A Development Method for the Conceptual Design of Multi-View Modeling Tools with an Emphasis on Consistency Requirements

Dissertation

zur Erlangung des akademischen Grades

Dr. rer. pol.

vorgelegt an der

Fakultät Wirtschaftsinformatik und Angewandte Informatik
der
Otto-Friedrich Universität Bamberg

von

Dominik Bork

vorgelegt:
29-04-2015
verteidigt:
13-07-2015

Gutachter:
Prof. Dr. Elmar J. Sinz
Prof. Dr. Dimitris Karagiannis

Acknowledgments

First of all I would like to thank Professor Elmar J. Sinz, my doctor father, for giving me the chance to start working as a research assistant at the University of Bamberg. Professor Sinz and the whole team established a community that allowed me to feel comfortable and to emerge both as a person and a researcher. The willingness to intensively discuss ideas and premature parts of this thesis together with the support in writing my first scientific papers considerably increased my confidence and motivated me to finish this thesis. Although I had to leave Bamberg, Professor Sinz insisted to further act as my doctor father and finish the project we started together. Albeit leaving Bamberg, Professor Sinz's interest in my thesis never abandoned. I cannot be thankful enough for that.

During my time at the University of Bamberg, several people helped me by establishing a comfort zone I was thankful to work in. Therefore, my special thanks go to all the colleagues at the research group systems development and database engineering, the forFlex research project members, and the members of the CEUS and the FlexNow teams. Additionally, I like to thank personally Felix Härer, Andreas Steffan, Michael Stretz, and Steffen Witt who have participated in the development of the first prototype of the SOM modeling tool. It was a pleasure and a great experience leading my first development team.

Moving to Vienna changed a lot. I was introduced to a new research group, met new colleagues and, from the very beginning, I was treated like a senior, equipped with challenging responsibilities and personal accountability. From the start I felt like a valuable member of the new team. I like to thank Professor Dimitris Karagiannis for immediately given me the chance to finish my thesis in Vienna - not even considering "no" for an answer. He perfectly knew how to motivate and challenge me to reach ever new goals, especially in moments I lacked confidence. One of my closest colleagues at the research group, and in the meantime a dear friend of mine I cannot thank enough is Hans-Georg Fill. He always took time discuss my ideas and thoughts. His value on a personal and scientific basis cannot be quantified. Moreover, I would like to thank Robert Andrei Buchmann who recently published first articles together with me. The discussions with him are unique and helped to concentrate on the core.

Another person I would like to thank is Wilfrid Utz. Wilfrid is an expert in developing modeling tools and conceptualizing modeling methods based on the ADOxx platform. He was of enormous help in developing the SOM and the MuVieMoT modeling tool. Both tools significantly improved due to his drive towards finding creative and smart solutions for challenging problems. Moreover, I would like to thank Valentina Tessa who helped me creating several icons for the MuVieMoT method.

Most importantly I want to thank my family, my girlfriend Melanie, and all my friends. They never complained about the scarce time I had left for them. This thesis is especially dedicated

to all my loved ones, who supported me in such manifold ways. Only because of their endless and unconditional faith in me I was able to finish this thesis.

Dominik Bork
Vienna, 31st July 2015

Contents

List of Figures

List of Tables

List of Abbreviations

ABL ADOxx Application Library

AD Activity Diagram

ALL ADOxx Library Language

ARIS Architecture of Integrated Information Systems

ASP Answer Set Programming

ATL ATLAS Transformation Language

BPEL Business Process Execution Language

BPMN Business Process Model and Notation

BPMS Business Process Management Systems

CIMOSA Computer Integrated Manufacturing Open System Architecture

COS Schema of Conceptual Classes

DD Diagram Definition

DEM Deductive Enterprise Model

DSVL Domain-Specific Visual Language

EER Extended Entity Relationship

EMF Eclipse Modeling Framework

EPC Event-Driven Process Chains

GEM General Enterprise Model

IS Information Systems

IAS Interaction Schema

IDEF Integrated DEFinition Methods

IMM Integrated Meta Model

ISO International Standards Organization

ITU International Telecommunication Union

MDA Model Driven Architecture

MDE Model Driven Engineering

MDD Model Driven Development

MDSE Model Driven Software Engineering

MEMO Multi-Perspective Enterprise Modeling

MOF MetaObject Facility

MVMUC Multi-View Modeling Use Case

OCL Object Constraint Language

ODS Object Decomposition Schema

ODM Ontology Definition Metamodel

OMG Object Management Group

OMI Open Models Initiative

OMiLAB Open Models Initiative Laboratory

OMT Object-Modeling Technique

OOSE Object-Oriented Software Engineering

OSM Orthographic Software Modeling

OWL Web Ontology Language

QVT Query View Transformation

RDF Resource Description Framework

RM-ODP Reference Model for Open Distributed Processing

SADT Structured Analysis and Design Technique

SOM Semantic Object Model

SERM Structured Entity Relationship Model

SUM Single Underlying Model

SysML Systems Modeling Language

TAS Schema of Task Classes

TDS Transaction Decomposition Schema

TES Task-Event Schema

TGG Triple-Graph Grammar

TOGAF The Open Group Architecture Framework

TOVE Toronto Virtual Enterprise

UML Unified Modeling Language

VOSE Viewpoint Oriented Systems Engineering

XPDL XML Process Definition Language

Frederic P. Brooks Jr.

The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions. This essence is abstract, in that the conceptual construct is the same under many different representations. It is nonetheless highly precise and richly detailed.
I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labour of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared to the conceptual errors in most systems. (BROOKS JR, 1995, p. 182)

---

1 Introduction

The design of complex systems like enterprise information systems, enterprise architectures, or software systems and corresponding architectures is of increasing complexity due to the ever increasing smartness and information processing nature of everything that is part of these systems. The Internet of Things, smart factories, smart products, cloud computing are just a few buzz words, one is confronted with in today's digital life. For the design, the analysis, and the development of these systems, modeling is of paramount importance. Modeling of those systems therefore has to deal with that increasing complexity, too. Hence, coping with the complexity by means of layering, decomposing, or partitioning of an overarching system specification into multiple parts is an inevitable requirement.

Concrete examples from the enterprise modeling and enterprise architecture domain may underpin that statement. Enterprise models are inherently layered or partitioned, mainly due to complexity management reasons. Established enterprise modeling methods and enterprise architecture frameworks use decomposition and abstraction by means of vertically and/or horizontally structuring an overarching enterprise representation along two or more dimensions into multiple facets, layers, or perspectives (cf. Semantic Object Model (SOM) (FERSTL AND SINZ, 2006, 2013), Multi-Perspective Enterprise Modeling (MEMO) (FRANK, 2002), Zachman framework (ZACHMAN, 1987), The Open Group Architecture Framework (TOGAF) (THE OPEN GROUP, 2011), ArchiMate (THE OPEN GROUP, 2012). Although the different approaches differ only slightly in many aspects, there is no common terminology used to describe and align them.

As the different decomposition approaches utilize different metaphors for decomposition, the nature of the relationships between the decomposed parts varies: a) they can be rather abstract/generic, typically mereological, cf. subprocesses part of higher level processes, as in the SCOR framework (COUNCIL, 2012), instance specifications or specializations, cf. the physical layer relative to the logical layer in Zachman's framework (ZACHMAN, 1987); b) they can also have richer semantics, cf. technology supports application, application supports business in ArchiMate (THE OPEN GROUP, 2012), or the Why-How-What-Who-Where-When inter-facet relations in Zachman's framework. Investigating the SOM enterprise modeling method, two concrete perspectives are given: *outside and inside a business system*; two abstraction layers are defined: *task level and resource level*; three model layers are distinguished: *enterprise plan*, *business process model*, and *resource layer*.

One widely accepted approach that is prominently utilized to decompose the complexity is *multi-view modeling* (CICCHETTI ET AL., 2011). Multi-view modeling enables system design and specification by decomposing an overarching system representation, i.e., a model of the system, into multiple interrelated views on that model (BROY, 2012). Each view concentrates only

on certain aspects of the whole system by intentionally omitting all others. However, the benefits coming from the application of multi-viewing come at a cost. As proposed by (JACKSON, 1990, p. 344), *"Having divided to conquer, we must then reunite to rule"*. The views generally are not isolated, i.e., independent from each other. More precisely, the views are overlapping by means of shared modeling concepts and/or semantics. *"Views of the same system, and viewpoints in general, are normally not entirely orthogonal, but have relations to each other. These relations are caused by overlap in the concerns that guide the viewpoint definitions, data which is shared over several views and through process constraints"* (PERSSON ET AL., 2013, p. 4). Integration of and consistency management between the views is still an open research issue (cf. HILLIARD (1999)). SINZ (1996) already stated that two major research gaps exist when it comes to multi-view modeling: 1) there is a lack of formalized specification of the modeling language provided by a viewpoint; and 2) even if there is a formalized specification of the viewpoint's modeling language, e.g., by means of a viewpoint meta model, there is a lack of formalized specification of the integration of multiple viewpoints.

When thinking of applying multi-view modeling by human beings, it must be clearly specified at which viewpoints which modeling operations can be triggered. Even more, it is of major importance to consider the effects on all viewpoints caused by the execution of a certain modeling operator on a certain viewpoint. *"... in fact, modifications operated within one view can have impacts on other views, often pertaining to the semantics of the considered domains, demanding a thorough specification of interplays between the different views"* (MIOTTO AND VARDANEGA, 2009). The effort of providing a comprehensive modeling operations specification in this context increases of course with the number of views considered.

Albeit multi-viewing has a long tradition in relational databases and requirements engineering, there is still no common understanding or terminology on a meta modeling level, defining the ways of a) specifying the general way of carrying out multi-view modeling, and b) conceptualizing multi-view modeling tools. When it comes to the development of multi-view modeling tools, one is confronted with multiple design decisions that need to be made, although the multi-view method seems to be specified thoroughly. Hence, there is a semantic gap between the method engineers understanding of a multi-view modeling method on the one side and the concerns of tool developers responsible for implementing a multi-view modeling tool on the other. Both domains, i.e., the conceptual domain of method engineering and meta models, and the technical realization domain of meta modeling based tool development platforms and conceptualization, need to be bridged.

Figure 1 illustrates the semantic gap between method engineers and tool developers. Method engineers utilize a *conceptual thinking* mindset, i.e., they decide on the relevant aspects of the reality and the appropriate concepts on meta level, codifying these aspects. By contrast, tool developers have a *design thinking* mindset. They rely on a given modeling method specification,

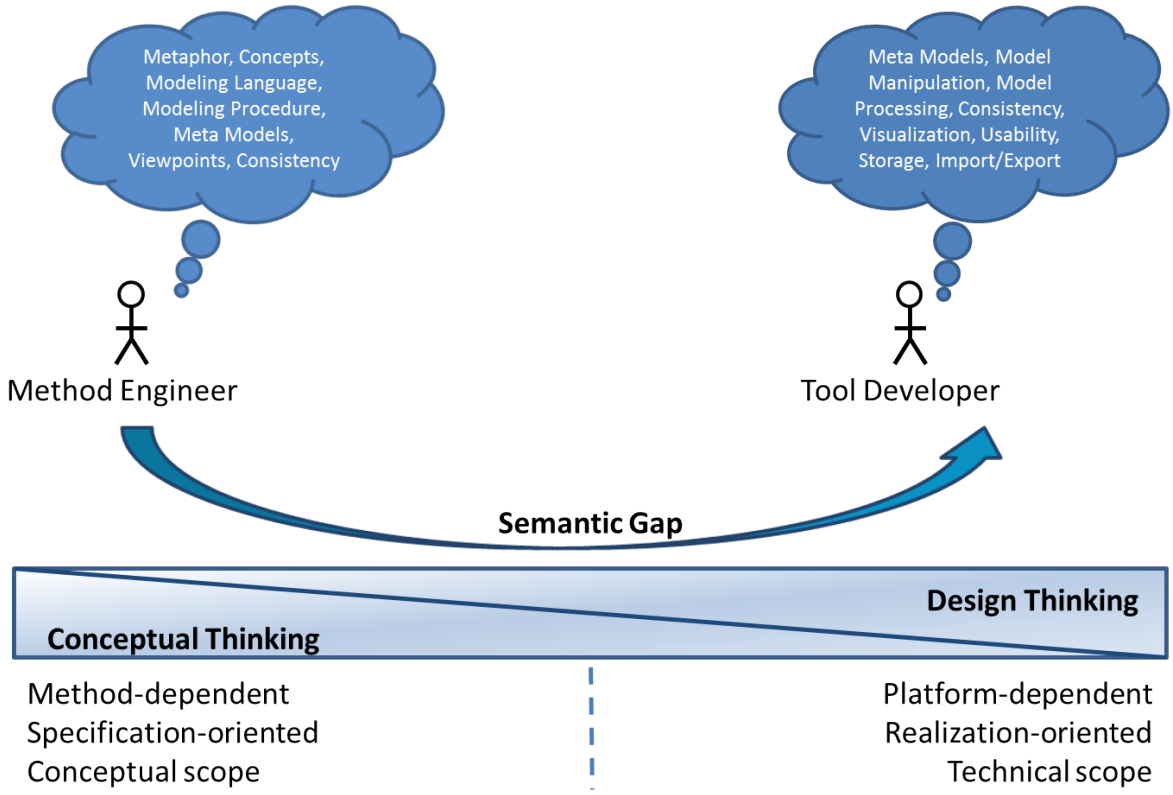


Figure 1: Bridging the semantic gap between method engineers and tool developers

independently of its completeness and degree of formalization, and think about how to design a technical realization of the specification by means of a modeling tool. A mapping of the conceptual world into the technical realization spectrum is performed by contrasting the requirements of the method engineer with the functionality of a certain tool development platform.

The thesis at hand contributes bridging that gap by means of proposing a conceptual modeling method called MUVIEMOT that is specifically designed to handle the specific requirements of the multi-view modeling domain and to cover both sides of the conceptualization spectrum. The MUVIEMOT modeling languages are coupled with each other by means of an accompanying procedural approach. Hence, the benefits of a procedural approach guiding the application of the approach is combined with the inherent benefits of conceptual models to capture requirements more precisely. Thereby, development errors based on inadequate or incomplete requirements specifications can be omitted (MAES AND POELS, 2007; MOODY, 2005). The MUVIEMOT approach aims at enabling method engineers in specifying a conceptual design for a multi-view modeling tool. With each step, the focus is shifting from a quite superficial but comprehensive abstraction level towards a more concrete multi-view modeling tool requirements specification, possibly aligned to a certain tool development or meta modeling platform. Approaches dealing with the integration of multiple (domain-specific) modeling methods are not considered in this

work (cf. KÜHN (2004); MIOTTO AND VARDANEGA (2009)).

Relationships between views and consistency requirements arising from them must be considered in the early conceptualization steps of modeling tool development. Conventional software engineering procedure models lack at covering the specifics of multi-view modeling methods. Moreover, the way how modelers interact with multiple views during the creation of a model is not covered in research until now; there is no comprehensive methodological support in the conceptual design of multi-view modeling tools. Like for embedded systems, *"there is an emerging need for a higher level modeling environment that facilitates efficient handling of this complexity"* (WOOD ET AL., 2008, p 1357). Meta modeling platforms are not convenient to capture the specifics of multi-view modeling methods. Moreover, such platforms only support development, not the conceptualization of modeling tools.

The aim of this thesis is therefore to help method engineers and tool developers in specifying the conceptual design of multi-view modeling tools. A conceptual design captures the requirements of a modeling method and maps it either to the functionality of a specific tool development platform or to the generic constituents of such platforms. MUVIEMOT is designed to provide a more suitable abstraction level, method engineers and tool developers feel comfortable with while answering the practical requirement of decomposition and integration when designing and developing multi-view modeling tools. The method is comprised with a modeling tool, developed with the ADOxx meta modeling platform. The tool allows efficient application of the method and model-driven development of multi-view modeling tools.

1.1 Motivation & Relevance

The research field of multi-view modeling and corresponding modeling tool support is of increasing interest in the information science and computer science research communities. This statement can be underpinned by searching for the terms 'multi-view modeling' in common scientific research databases. There is a clear, linear increase of the number of scientific publications that deal with multi-view modeling, multi-view modeling methods, multi-view modeling tools, multi-view consistency, and model-driven development in the last 10 to 20 years. Moreover, several scientific workshops, e.g., *Methodical Development of Modeling Tools (ModTools)*[1] and tracks or mini-tracks at international conferences have been established in the community, specifically dealing with research questions on the conceptualization of modeling tools, consistency in multi-view models, and the utilization of multi-viewing in manifold domains.

Following the introductory statements, the increasing complexity of e.g., business markets, enterprise models, enterprise architectures, or business application systems makes it an in-

[1] The workshop methodical development of modeling tools has emerged from a workshop concentrating on the German speaking area called *Methodische Entwicklung von Modellierungswerkzeugen (MEMWe)*. This workshop series originated in 2009 at the *Informatik* conference.

evitable prerequisite to utilize multi-view modeling for these domains. It is necessary to support human beings with modeling tools, specifically designed to cope with this complexity by providing multiple views and keeping them consistent. Consequently, there is a multitude of domains that utilize multi-viewing, e.g., architecture description (IEEE, 2011), cyber-physical systems (BHAVE ET AL., 2011; BROMAN ET AL., 2012), integrated product-process design (DEMOLY ET AL., 2010), enterprise modeling (FERSTL AND SINZ, 2013; FRANK, 1994; ZACHMAN, 1987), enterprise architecture standards like *ArchiMate* (THE OPEN GROUP, 2012) or The Open Group Architecture Framework (TOGAF) (THE OPEN GROUP, 2011), or software and systems modeling (OBJECT MANAGEMENT GROUP (OMG), 2012b; SHAH ET AL., 2010; MUNKER ET AL., 2014).

The thesis at hand therefore tries to contribute to a research field with an estimated high impact in the near future that is currently at its early stages. This statement is underpinned by an recently published position paper that emphasizes on the impact and the open research issues in the field of modeling business information systems (FRANK ET AL., 2014). The authors state that *"Modeling research is well advised to take into account future developments, to create models, languages, and methods that can be adapted to changing conditions"* (FRANK ET AL., 2014, p. 3). The authors also emphasize on the importance of appropriate tool support by stating *"We must assume that in the future end users will increasingly design and manage models. Modeling tools and corresponding languages developed according to requests from end-users should, therefore, move into the focus of research"* (FRANK ET AL., 2014, p. 3).

Overall, the analysis of the related work revealed the following fundamental shortcomings and inadequacies for multi-view modeling:

- Prior to the development of multi-view modeling tools, a fundamental analysis must be performed in order to gain an overview of the scenario, the modeler is placed within while interacting with multiple views on a system. No approach considers this superficial but important task.

- Common requirements specification languages lack at supporting the specificity of multi-view modeling methods and corresponding tools.

- Consistency between multiple views plays a paramount role in terms of model quality, modelers satisfaction, and machine processing of the models. Therefore, an emphasis must be on the specification of viewpoint dependencies, enabling the development and utilization of consistency-preserving multi-view modeling tools.

- The actual way of carrying out multi-view modeling is not targeted at all in research up to now, e.g., interactions between a modeler and the multiple views of a model, constraining modeling operations to certain viewpoints.

The thesis at hand contributes filing these research gaps. The content of the thesis builds on a set of research question, introduced in the following.

1.2 Research Questions

The overarching research question this thesis tries to answer is how to comprehend the specific characteristics of a multi-view modeling method into a conceptual design of a multi-view modeling tool in an intuitive and efficient manner, utilizing a suitable abstraction level. In the following, this research question is decomposed into more precisely formulated research questions, establishing the building blocks of this thesis.

RQ I: What is the theoretical and conceptual foundation of multi-view modeling and multi-view modeling methods?

One pillar of this thesis is an extensive theoretical research trying to obtain an overview of existing approaches and theories in the different ways of carrying out multi-view modeling. This research question is motivated by the fact, that multi-view modeling, in all its derivations, is applied in a wide range of disciplines and with different understandings of what actually multi-view modeling is. There is neither a common denomination nor a common understanding between the different derivations. Based on the findings, an abstract, solid, and methodically sound characterization of multi-view modeling and multi-view modeling methods from a meta modeling perspective shall be developed. This definition shall consolidate the backbone for further investigations.

RQ II: Which consistency issues must be considered when developing a multi-view modeling tool?

With the usage of multiple views, altogether realizing a model of some aspects of the reality, consistency issues are inherently given and must be regarded seriously. The acceptance and efficient utilization of multi-view modeling methods, and corresponding tools, considerably depends on the availability of consistency-preserving concepts, mechanisms and techniques. Whereas for single-view modeling an experienced modeler may be able to cope with temporarily inconsistencies by hand, the complexity of multi-view modeling forces the development and utilization of technical solutions towards consistency management. Work on this research question implies performing a literature review in order to find and compare existing approaches on consistency management. The specifics of multi-view modeling methods will furthermore require the definition of additional, particularly dedicated consistency classes and accordingly designed mechanisms.

RQ III: What influence does the availability of an integrated meta model have on the conceptual design of a multi-view modeling tool?

One selective differentiator for comparing multi-view modeling methods is the availability of an integrated meta model. However, yet there is no research published about the influence of an integrated meta model on the requirements of a multi-view modeling tool. Especially the consistency of the views must be investigated in both scenarios: (1) with an integrated meta model, and (2) without an integrated meta model. The thesis should provide a clear distinction of these cases and provide discussions and solutions for both cases. This also enables the results of this thesis to be more comprehensive and applicable in a broader context. However, the focus is on the case with an integrated model.

RQ IV: How can the conceptual design specification of multi-view modeling tools be supported by a methodical approach?

The main goal of this thesis is to develop a methodological approach that contributes bridging the gap between a multi-view modeling method on the one side and the conceptual design of an appropriate multi-view modeling tool on the other. Simultaneously, the approach aims at bridging the different abstraction levels of method engineers and tool developers (cf. Figure 1). Therefore, a sequential approach should be invented, breaking down the complexity of the conceptualization process into manageable steps. An emphasis of the hereby created conceptual design should be on the specific characteristics of multi-view modeling methods and their codification by means of tool requirements. This research question should also cover a discussion about the shortcomings of conventional software engineering and modeling tool development approaches when applying them in the development of multi-view modeling tools.

RQ V: What are the benefits of formalized modeling method specifications?

The development of modeling tools has two points of origin. On the methodical side a modeling method and on the technological side a tool development environment. Tool developers need precise information about the modeling method in order to be able to develop a comprehensive and efficient modeling tool based on the functionality given by the development environment. However, most modeling methods are specified on a more informal level, i.e., using natural language specifications for several parts of the method, e.g., notation, semantics, and modeling procedure. Part of this thesis is therefore to investigate how introducing formalization can improve the accuracy of modeling method specifications towards an unambiguously and inter-subjectively understandable level. The availability of formal-

ized specifications also fosters the processing of the created models by computer systems. Consequently, benefits and possibilities of formalized modeling method specifications shall be identified.

RQ VI: How can a graphical modeling method and a corresponding modeling tool ease the application of the method?
The developed method has no means on its own and should not be limited to the use cases described in this thesis. The method should be designed to be generally applicable on a broad basis, i.e. for arbitrary multi-view modeling methods, either with or without an integrated meta model. In order to ease the use of the developed approach, a modeling tool supporting the different steps of the approach in an intuitive and model-driven way shall be realized. Moreover, initial transformations between adjacent steps of the method shall be provided. Hence, the steps of the method shall be supported by specifically designed graphical modeling languages, integrated in a common modeling tool environment that is freely available.

The aim of the MUVIEMOT method is to lower the level of abstraction by grounding the multi-view concept in concrete concerns of modeling tool requirements in a meta modeling context. Method engineers and tool developers may consider the approaches hereby introduced in order to answer the practical requirement of decomposition and integration when designing and developing multi-view modeling tools.

1.3 Research Approach & Research Procedure

In the last couple of years a lot of researchers have participated in discussions on the value of design-oriented research as a complement to conventional, e.g., behavior-oriented, research approaches. The discussion is not new (cf. Susman and Evered's discussion on action research in 1978 (SUSMAN AND EVERED, 1978)). Nunamaker et al. and Walls et al. already discussed in the 1990s the benefits of considering systems development as an integral part of a multimethodological approach to information systems research (NUNAMAKER ET AL., 1990, p. 94), (WALLS ET AL., 1992). Behavioral science, originating from natural sciences, *"seeks to develop and justify theories (i.e., principles and laws) that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems"* (HEVNER ET AL., 2004, p. 76). Design-oriented research on the other hand, originating from an engineering background (SIMON, 1996), *"seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished (*DENNING*, 1997;* TSICHRITZIS*, 1997)"* (HEVNER ET AL.,

2004, p. 77). March and Smith even argue that, *"both design science and natural science activities are needed to insure that IT research is both relevant and effective"* (MARCH AND SMITH, 1995, p. 251).

The German-speaking research community picked up the discussions in the Anglo-Saxon area. In a memorandum on design-oriented information systems research, a group of 10 distinguished scientists emphasized on establishing a pluralistic view in the information science discipline, i.e., the value of innovative solutions for known problems aside of behavior-oriented research that provides *"statistically evidence of empirically identified characteristics"* (ÖSTERLE ET AL., 2010, p. 8). The goal of the initiative was to emphasize on the benefits and value of rigorously performed design science research and the contribution it can make to the scientific community. Worse acceptance rates of design science research articles initially preceded and motivated this discussion. Moreover, the authors were afraid on the influence of the trend towards behavior-oriented research at universities and young scientists. The memorandum has gained more impact as, besides the authors, 111 full professors from the German-speaking and Scandinavian scientific community supported it by signing the principles specified therein (ÖSTERLE ET AL., 2010).

According to Österle et al., rigorously performed design science research needs to account for these four principles (ÖSTERLE ET AL., 2010, p. 9):

- *Abstraction:* Each artifact must be applicable to a class of problems.
- *Originality:* Each artifact must substantially contribute to the advancement of the body of knowledge.
- *Justification:* Each artifact must be justified in a comprehensible manner and must allow for its validation.
- *Benefit:* Each artifact must yield benefit - either immediately or in the future - for the respective stakeholder groups.

According to the memorandum, design science research should ideally follow an iterative process, consisting of the sequence of these four steps (ÖSTERLE ET AL., 2010, p. 9): *Analysis*, analyze the specific problem situation, define research objectives and establish a research plan; *Design*, an artifact should be created and evaluated by means of comparing it to existing alternatives; *Evaluation*, rigorously evaluate and review the artifact against the identified research objectives; and *Diffusion*, the results of the design science research should be positioned as a contribution to the scientific and/or practice community by means of scientific papers and instantiations in companies, respectively.

The discussion on the value of design science research and the possibilities to publish high-quality design-oriented research papers at important international conferences and journals is

still ongoing. BICHLER (2014) lately emphasized that there is still a lack of numerous design-oriented research papers accepted at the most important and influential outlets. So ten years after the highly recognized paper of Alan Hevner (HEVNER ET AL., 2004), the scientific community has not evolved as expected.

The main goal of this thesis is the development of a methodical approach that is applicable when conceptually designing multi-view modeling tools. Therefore, the contribution of this thesis is an artifact that is aimed to be on an abstract level, allowing its non-restrictive application to arbitrary multi-view modeling methods. The artifact is evaluated with an illustrative scenario of the enterprise modeling domain, nevertheless it is aimed to be generally applicable, i.e., domain-independent.

The Design Science Research Methodology

Picking the right methodology for applying design science is a tough choice due to the numerous published theoretical approaches, e.g., (DAVISON ET AL., 2004; SEIN ET AL., 2011; WIERINGA AND MORALI, 2012; WIERINGA, 2012). Notably, most of these approaches originate from the Anglo-Saxon area, contributing to the design science research community with theoretical, conceptual or mereological frameworks. Those works propose how design science research should be conducted and evaluated.

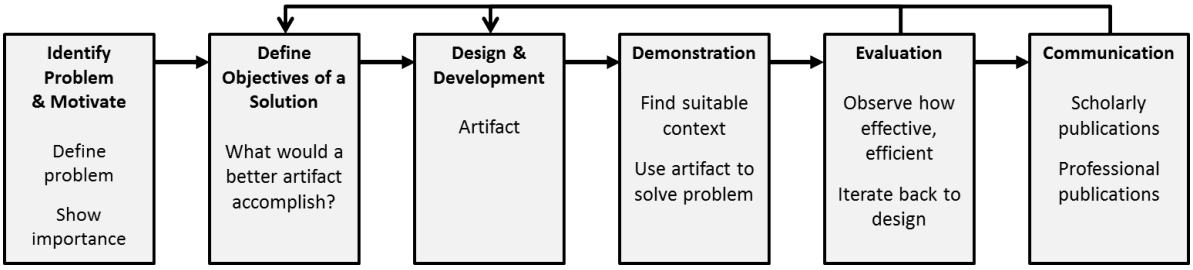


Figure 2: Design Science Research Methodology process model (PEFFERS ET AL., 2007, p. 54)

The *Design Science Research Methodology* proposed by Peffers et al. has significantly influenced the design science research community. It convinces through its simplicity and general applicability. By the end of February 2015, the paper introducing the framework has been cited 1190 times according to Google Scholar[2]. It is more specific compared to the four fundamental principles of design-oriented research proposed by ÖSTERLE ET AL. (2010). The Design Science Research Methodology process model, illustrated in Figure 2, comprises the following steps (PEFFERS ET AL., 2012, p. 48ff.):

1. **Identify Problem & Motivate:** The first step of the methodology suggests to identify the problem, decompose it into atomic aspects, and use the collection of problems to

[2] Google Scholar, `http://scholar.google.com/`, last checked: 2015-02-25

elaborate on the value of a proposed solution. This added value should foster motivation of: i) researchers, responsible for carrying out the design science research, and ii) the scientific community in accepting the relevance and contribution of the research.

The *Identify Problem & Motivate* step of the methodology is covered in this thesis as follows: Starting with a motivation in section 1.1, already on the first pages of this thesis a clear motivation statement is given. This motivation is underpinned by the analysis of related approaches and the identification of shortcomings and/or inadequacies of them in section 3.

2. **Define Objectives of a Solution:** The second step of the methodology compares the identified problems with the state of the art, hereby defining concrete objectives of a desired solution. This objectives might be qualitative or quantitative by nature, depending on the artifact to be developed and the context it should be placed in. The objectives should be rationale, i.e., they should be achievable with the current knowledge bases and possibilities.

   The *Define Objectives of a Solution* step is covered in this thesis essentially by an overarching requirements specification in section 6.2. The requirements build on three pillars: 1) the current state of literature on multi-view modeling and the development of corresponding tool support (discussed in section 2 and section 3); 2) the investigation of capabilities and functionality of existing approaches in supporting method engineers in codifying the specifics of multi-view modeling methods (discussed in section 3); and 3) the lessons learned and inadequacies experienced during the development of multi-view modeling tools from scratch.

3. **Design & Development:** The core of the third step of the methodology covers the design and the development of the artifact. Starting from design and functionality specifications, the (technical) realization of the artifact is focused. Artifact types are methods, models, constructs, and instantiations.

   The *Design & Development* step of the methodology is covered by the main contribution of this thesis, the MUVIEMOT modeling method and its supporting modeling tool. Hence, the thesis is considered as a contribution of the type *method*. MUVIEMOT is described thoroughly in section 6. As additional contributions to the design science research community can be considered the following parts of this thesis: The analysis framework presented in section 4, and the MUVIEMOT modeling environment with the functionality for model-driven development of multi-view modeling tools described in section 8.

4. **Demonstration:** This step covers the instantiation of the realized artifact in real-world or artificial situations by means of e.g., case studies, illustrative scenarios, experimentations,

or simulations.

The *Demonstration* step of the methodology is covered in this thesis by a comprehensive illustrative scenario performed with the MUVIEMOT modeling method. Section 7 describes the application of all steps of the method to the Semantic Object Model (SOM) enterprise modeling method in order to specify a conceptual design for a SOM multi-view modeling tool.

Considering the analysis framework, its feasibility is demonstrated in section 4.2 by an analysis of several enterprise modeling methods, hereby investigating on the different levels of formalization in the modeling methods' specifications. The MUVIEMOT modeling tool has been used in the same illustrative scenario as the MUVIEMOT method, hence producing a conceptual design for SOM by means of graphical models and transformation of these models into an initial multi-view modeling tool, presented in section 8.

5. **Evaluation:** This step observes and measures *"how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration"* (PEFFERS ET AL., 2007, p. 56). The scientific community emphasizes on the importance of rigorously performed and evaluated design science. *"It is the rigor of constructing IT artifacts that distinguishes Information Systems as design science from the practice of building IT artifacts"* (IIVARI, 2007, p. 50). Venable argues, that *"Evaluation is what puts the "Science" in "Design Science". Without evaluation, we only have an unsubstantiated design theory or hypothesis that some developed artifact will be useful for solving some problem or making some improvement"* (VENABLE ET AL., 2012, p. 425).

   Although there is a general agreement on the importance of evaluating designed artifacts, the choice of the most appropriate evaluation technique is not trivial. VENABLE ET AL. (2012) identified five purposes of evaluation and three different goals of performing an evaluation. (HEVNER ET AL., 2004, p. 86) identified five design evaluation methods. PRIES-HEJE ET AL. (2008) provided a design science evaluation framework by combining naturalistic/artificial and ex ante/ex post evaluation strategies. The list can be continued much longer.

   The *Evaluation* of MUVIEMOT follows consequently the guidelines of an analysis of performed evaluation types applied to certain artifact types in the design science research community, published by partly the same authors that developed the Design Science Research Methodology (PEFFERS ET AL., 2012). Peffers et al. identified, that the most applied evaluation techniques for the artifact type 'method' were technical experiments, case studies, and illustrative scenarios. In case of modeling methods, hence relevant for the MUVIEMOT modeling method, *illustrative scenarios* are favorable as they describe

the *"application of an artifact to a synthetic or real-world situation aimed at illustrating suitability or utility of the artifact"* (PEFFERS ET AL., 2012, p. 402).

Following this argumentation, the illustrative scenario of the MUVIEMOT method in section 7 describes utility of the method using an application of the method to the SOM enterprise modeling method. Notably, considering the discussion of the degree of fulfilling the objectives and requirements can be considered a qualitative evaluation according to the former evaluation requirement, described in the original publication of the Design Science Research Methodology (PEFFERS ET AL., 2007). Moreover, the results of utilizing MUVIEMOT were compared to the effort of developing a SOM multi-view modeling tool from scratch.

6. **Communication:** The last step of the methodology covers the diffusion of the research results in the scientific community e.g., by scientific papers, demonstrations, public discussions. The knowledge gained during the research process should be made publicly available. The publication should be structured in an adequate way, e.g., by referring to the first five steps of the Design Science Research Methodology. Hence, not only the artifact should be described, but also the targeted problems, the relevant related approaches, the design, and the evaluation.

   The *Communication* step of the methodology is covered in three ways: First, several parts of this thesis have been published in respective international journals or international scientific conferences. Appendix C provides a complete list of the published articles and outlets. Second, the thesis at hand provides a contribution to design science research as it comprehensively describes the whole research process. Third, the tools developed while working on this thesis, i.e., the SOM multi-view modeling tool and the MUVIEMOT modeling environment, are publicly and free available on the Open Models Initiative Laboratory (OMiLAB) webpage[3].

1.4 Outline

Following this introduction, section 2 defines the methodological and conceptual foundations of this thesis. More precisely, the aim is on introducing a common understanding of the concepts of *modeling*, *modeling methods*, *meta modeling*, *multi-view modeling*, *conceptual modeling* and *model-driven development*. This foundation is essential for the classification of the forthcoming contributions of this thesis.

In section 3 related work on the conceptual design and the development of multi-view modeling tools is discussed with the aim of distinguishing the contribution of this thesis to existing

[3] The OMiLAB webpage: `http://www.omilab.org/`, last checked: 2015-04-01

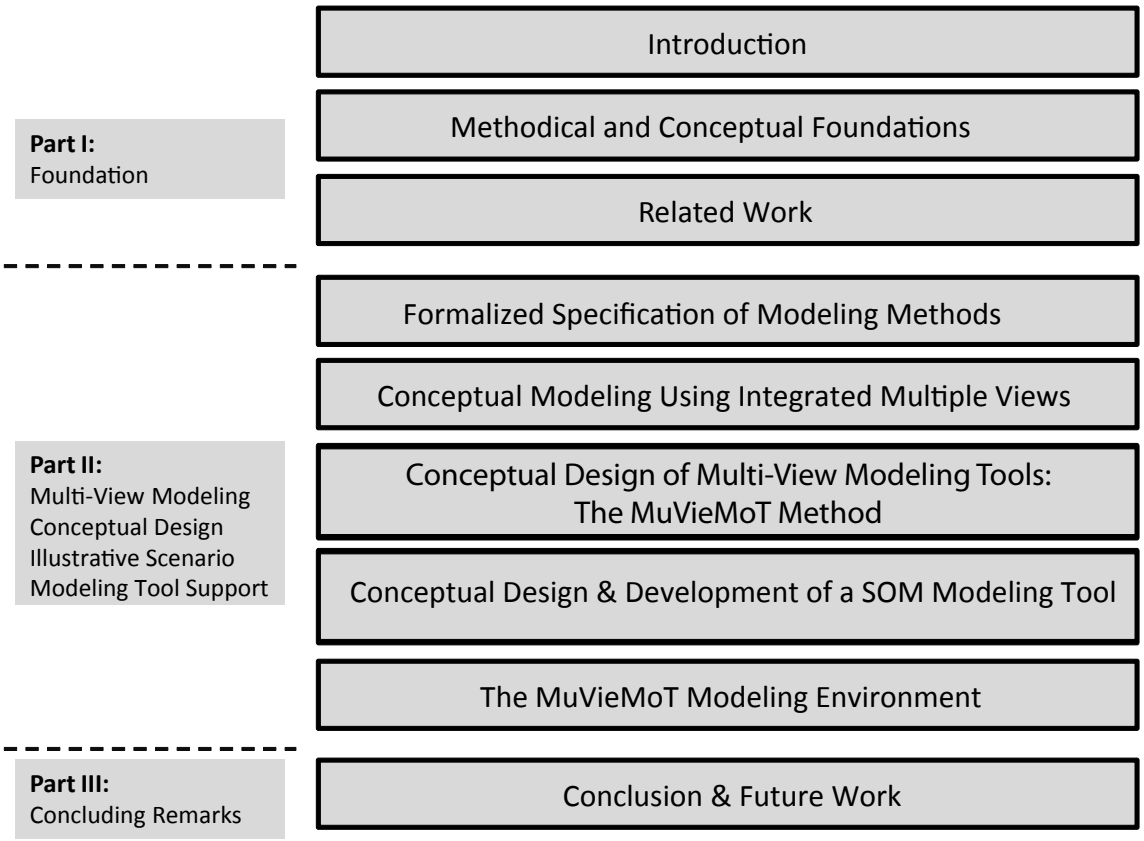


Figure 3: Structure of the thesis

works of other authors. Moreover, this section serves as a source of motivation for the development of the MuVieMoT method by pointing to the identified inaccuracy and inadequacy of the existing approaches.

In section 4 the benefits, drawbacks and possibilities of a formalized modeling method specification are discussed. The discussion is structured by applying a specifically developed analysis framework to a set of modeling methods, hereby investigating the degree of formalization the different modeling methods provide. In a concluding section, the benefits of such formalization are pointed to with concrete examples of the investigated methods.

In section 5 the characteristics of integrated multi-view modeling methods and multi-view modeling are defined. An emphasis is on a proper specification of the fundamental terminology this thesis builds upon, comprising the relationship types between views and the different ways of carrying out multi-view modeling. A generic dichotomy of applying multi-view modeling is presented: *multi-view by design* and *multi-view by generation*.

In section 6 the MuVieMoT method is introduced. First, requirements for a method aiming at developing a conceptual design for multi-view modeling tools are gathered. Subsequently, the steps of the MuVieMoT method are introduced by defining their purpose and, if given, the specifically designed modeling language. The MuVieMoT modeling procedure and the

relationships between the modeling languages are defined accordingly. A first evaluation of MUVIEMOT by means of relating the requirements to the constituents of the method is performed. Moreover, the constituents of MUVIEMOT are contrasted with the research questions. Lastly, a SWOT analysis of the method is performed.

Section 7 describes an illustrative scenario of applying the MUVIEMOT method to a concrete multi-view modeling method. Hence, *Modeling Scenario*, *Modeling Language*, *Modeling Procedure*, *Viewpoint Dependencies*, and a *Conceptual Design* of a consistency-preserving multi-view modeling tool for the Semantic Object Model (SOM) method are described. This section provides a thorough introduction to the SOM method and to the ADOxx meta modeling platform. ADOxx has been used to demonstrate the feasibility of the conceptual design by means of a technical implementation. Consequently, the SOM modeling tool is described and an evaluation is presented, discussing lessons learned and benefits of applying the MUVIEMOT method.

The focus of section 8 is on the designed and implemented MUVIEMOT modeling environment, realized on the ADOxx meta modeling platform. It allows the model-driven application of the MUVIEMOT lifecycle by means of conceptual models, specifically designed according to the needs of the MUVIEMOT steps. While introducing the tool, the SOM case study is realized a second time, this time showing all graphical models created with the MUVIEMOT tool. A major benefit of the tool is not only the methodological support for MUVIEMOT but also the model-driven development of multi-view modeling tools. The created conceptual design can be transformed into an initial consistency-preserving multi-view modeling tool implementation in ADOxx. In the evaluation of the tool, such a model-driven development process is described for a SOM business process modeling tool.

In section 9 the thesis is concluded. The major contributions of the thesis are summarized, lessons learned are discussed, limitations of the current version of the MUVIEMOT method are clarified and possible directions for future research are outlined.

In Appendix A further information on the SOM modeling tool's illustrative scenario, release notes, and acknowledgments is given.

In Appendix B further information on the development of the MUVIEMOT modeling environment, comprising release notes and acknowledgments, is given.

2 Methodical & Conceptual Foundations

The foundations of this thesis are based on one side on the theory of modeling, meta modeling, and multi-view modeling. On the other side, concepts for defining consistency, for formalizing method specifications, and for designing and developing modeling tools, all in the context of a multi-view modeling method, are relevant. The following section therefore provides a sound theoretical and methodical basis, the thesis is grounded on.

The following section discusses the foundations of modeling, modeling methods, and meta-modeling (section 2.1). Multi-view modeling (section 2.2) is then introduced with an emphasis on relationships between views and consistency issues raised by them. Section 2.3 introduces conceptual modeling. Finally, section 2.4 describes the foundations of model-driven development.

2.1 Modeling Foundations

Modeling has emerged from being on the fringes to one of the cornerstones of today's information science education. Modeling, and therefore models, are not only used in education, they play a mature role in industrial practice as well. Enterprises use models e.g., to analyze their (business) processes, to simulate alternative production lines, to visualize their organizational hierarchy, and in a lot of application areas more. Furthermore, models play also a very important role in the fields of Model Driven Architecture (MDA) (MILLER AND MUKERJI, 2003; FETTKE AND LOOS, 2003) and Model Driven Engineering (MDE) (SCHMIDT, 2006; KENT, 2002) as they are the basis for the subsequent semi-automated development of architectures and application systems. The most important roles of models in information science therefore can be classified into two classes: (1) analysis of existing systems, and (2) development of new systems. This classification is of course neither comprehensive nor disjunctive, as models can act as a description of an as-is situation in the first place, and, afterwards as a description of a to-be situation. Furthermore, models can be used in a divers set of situations not regarded in depth in this thesis (e.g., as experimental models, for simulation, for verification and validation).

2.1.1 Models & Modeling

As the complexity of information systems, enterprises, and software architectures raises steadily, human beings are forced to use abstraction in order to retain an overview of the investigated or designed sub-area of the real world. It is common practice to create models when analyzing complex real world or artificial systems. A model is *"a representation of either reality or vision"* (WHITTEN ET AL., 2004, p. 187). In some domains, the investigated system is referred to as a system under study (SEIDEWITZ, 2013). Stachowiak introduced these three criteria to

describe the nature of modeling on a theoretical basis (STACHOWIAK, 1973, p. 131-133):

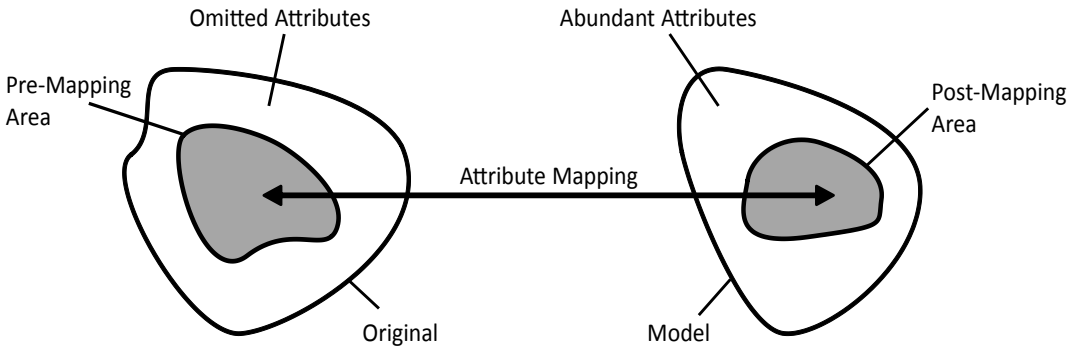


Figure 4: Theoretical definition of modeling (STACHOWIAK, 1973, p. 157)

- Mapping
  Models always represent something, i.e., they rely on a concrete real or artificial artifact which itself can be a model. Models therefore act as a mapping or representation of this artifact (in the following related to as "original"). *Stachowiak* defines both, model and original as attribute classes. The mapping relation between an original artifact and a model can therefore be regarded as the mapping between attributes of the original to the model (in the sense of a mathematical mapping in set theory).

- Reduction
  Models generally do not include all attributes of the original artifact, i.e., they include only a subset of attributes that is dedicated to the need of the modeler and/or the model user. Therefore models provide a certain level of abstraction.

- Pragmatism
  The last criteria specifies, that during the description of what constitutes a model, multiple questions should be considered: *what* is the model built of (i.e., the original); *when* is it being built; *for whom* is it built; and *which purpose* does it serve.

The mapping of some part of the real world by means of creating a model of this part needs to be discussed in more detail. As mentioned earlier, Stachowiak describes both, original and model as attribute classes. During the mapping process, those attributes play an important role as they (1) can be integrated in the mapping, (2) can be omitted in the mapping as they don't matter according to the current pragmatism of the model (*omitted attributes*), or (3) they are added to the model although they have no corresponding attribute in the original, i.e., because of technical or economic issues, or because of some limitations of the mapping relation according to the pragmatism of the model (*abundant attributes*). Figure 4 illustrates the mapping between the original and the model.

A more formal definition of the term model and modeling is given by Ferstl and Sinz. The authors define a model informally as *"a system, that represents an other system by a goal-oriented manner"* (FERSTL AND SINZ, 2013, p. 137ff.). A model therefore consists of three constituents: an *Object System* S_O, a *Model System* S_M, and a *Mapping Function* $f : V_O \longrightarrow V_M$. The mapping function relates system components V_O of the object system S_O to the system components V_M of the model system S_M. Structural and behavioral conformance between S_O and S_M depend on whether the mapping function f is homomorphous or isomorphous, respectively. The *Object System* determines an excerpt of the reality the modeler is currently interested in, e.g., the business processes of an enterprise if a business process model is to be created. The *Model System* is a formal specification of the *Object System*, e.g., a data model, a business process model, an organizational model, created using the concepts formally defined in the *Meta Model*. Figure 5 illustrates the definition of modeling by Ferstl and Sinz.

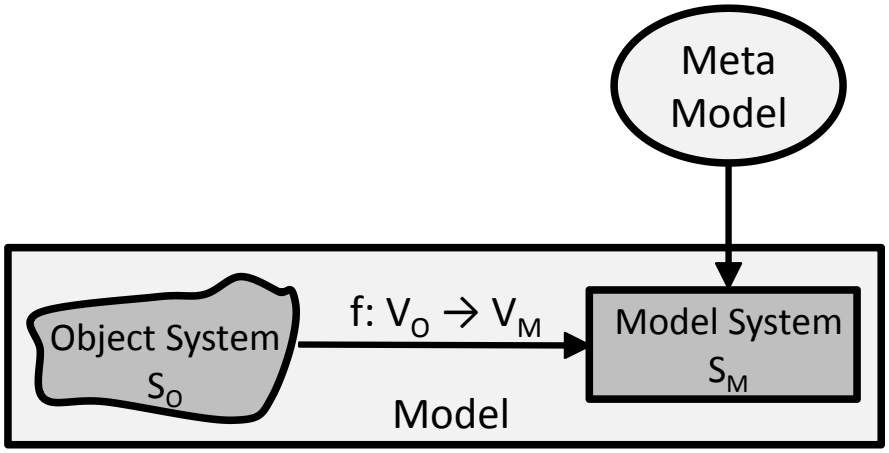


Figure 5: Formal definition of modeling (FERSTL AND SINZ, 2013, p. 129)

Summarizing the definitions above, modeling can be stated as describing some aspects of the reality in more detail while ignoring and abstracting from others, thereby serving the purpose of the modeler and/or model user. Modeling can be used in order to break down the complexity of the reality and therefore provides a basis for an inter-subjectively understandable knowledge base. The definitions of model and modeling do concentrate on the mapping (function) between some real world phenomenon and a model. However they do not consider one vital aspect that is the basis for each modeled artifact: the modeler who perceives the real world, delimits the parts of the real world to be considered, and finally creates the model of that part of the real world by instantiating the mapping function. Ferstl and Sinz therefore introduce a complementary definition by reverting to a constructivist understanding of modeling (FERSTL AND SINZ, 2013, p. 138f). Figure 6 illustrates this constructivist understanding of modeling.

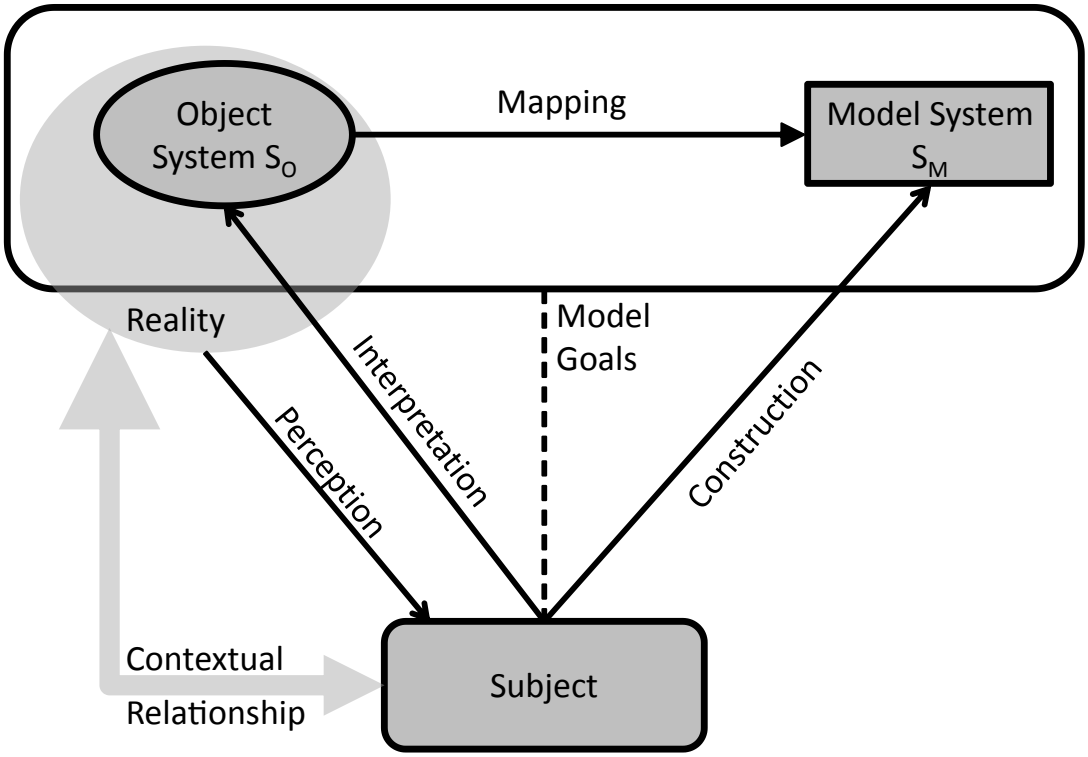


Figure 6: Constructivist understanding of modeling (FERSTL AND SINZ, 2013, p. 130)

In the center of the constructivist understanding of modeling stands a *Subject* i.e., a human being in the role of a modeler who is driven by certain goals while building models. Bézivin also emphasized on the goal-driven aspect by stating, *"A model is a simplification of a system built with an intended goal in mind"* (BÉZIVIN AND GERBÉ, 2001, p. 274). The subject also stays in a contextual relationship to the reality, causing a subjective perception of the reality and interpretation of the object system. This perception and interpretation, driven by the goals pursuit, results in the models created by a human being.

Consequently, the process of modeling is always a very subjective matter, it is therefore worthwhile to reduce the subjective influence in modeling (e.g., by providing meta models, metaphors, process models) and providing an inter-subjective understandable model. The inter-subjective aspect comes with the usage of a commonly agreed modeling language, defining the general way of investigating the reality and building a model according to a modeling language. The comprehensive specification of a modeling language together with its application in order to build valid models are subsumed under the terminology of a **modeling method**, which will be introduced in the following.

2.1.2 Modeling Methods

Having defined the concepts of models and modeling, the following section uses those concepts and expands them to the definition of a modeling method. Again, two approaches to define the constituents of modeling methods are presented and discussed afterwards. However, before the constituents of modeling methods are discussed in detail, a short look at a definition of an information systems method may be given: *"a coherent collection of concepts, beliefs, values and principles supported by resources to help a problem-solving group to perceive, generate, assess and carry out in a non-random way changes to the information situation"* (LYYTINEN, 1987; AVISON AND WOOD-HARPER, 1990). This general definition already takes into account the important mapping between a certain problem domain and specifically designed methods, aiming to be suitable for that problem domain. There is no such thing as a modeling method that is generally applicable in any domain. Standards like the Business Process Model and Notation (BPMN) (OBJECT MANAGEMENT GROUP (OMG), 2011a) for business process modeling or the Unified Modeling Language (UML) (OBJECT MANAGEMENT GROUP (OMG), 2011d) for software systems modeling lack expressiveness when it comes to mapping the specifics of a certain domain. Hence, the modeling method must be chosen in accordance to the investigated domain and the information needs of the model users that should be met. In order to decide between the rich set of available modeling methods, it is important to compare the methods on a conceptual basis. The usage of the subsequently introduced approaches yields a comprehensive description of modeling methods and therefore fosters comparison of them.

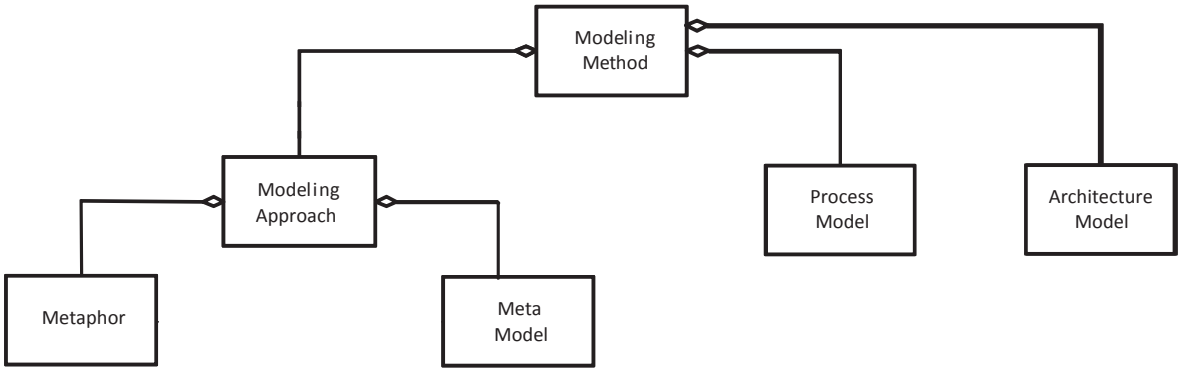


Figure 7: Components of modeling methods according to Ferstl and Sinz

According to Ferstl and Sinz (FERSTL AND SINZ, 2008, p. 130f.), constituents of a *Modeling Approach* are a *Metaphor* and a *Meta Model* (cf. Figure 7). Modeling approaches can be extended to the definition of a *Modeling Method* by incorporating an *Architecture Model* and a *Process Model* (BORK AND SINZ, 2013). These constituents of a modeling method are now introduced in more detail:

- Metaphor
  A metaphor, in the common understanding, is a figure of speech, used as an analogy to describe a certain situation or object by referring to a generally known but normally not related situation or object, e.g., something is described as being "crystal clear". In the context of modeling, a metaphor defines the general way of perceiving the real world and delimiting the part of the real world that should be considered within the model. The metaphor also defines the way of building this part of the real world by means of a conceptual model.

- Meta Model
  A central aspect of a modeling method is the modeling language (i.e., objects and relations with attributes). The elements of a modeling language that can be used to create valid models are formally specified in a meta model. The meta model also constrains the relations between the language's elements. Using the metaphor, meta models can be described as the general construction plan of all model instances that can be created using the modeling method. According to (FERSTL AND SINZ, 2008, p. 131), a meta model also defines the semantics (i.e., the meaning) of all elements. The semantics of the elements, and therefore the semantics of the modeling language, has to be aligned to the metaphor of the modeling language.

- Architecture Model
  The architecture model structures the model by defining model layers, sub-models, viewpoints, and perspectives, therefore breaking down the complexity of both, the model as a whole and the task of creating valid models. Consequently, architecture models are vital for the specification of large models, e.g., enterprise models and enterprise architectures.

- Process Model
  The process model is used to describe the steps to be performed by the modeler in order to build valid models. The aspects defined in the architecture model serve as an input for the process model, which uses this input to define a sequence of modeling steps, e.g., fist create model A, transform model A into model B, then refine model B. Hence, the process model guides the modeler during the application of a modeling method.

A different framework to describe modeling methods comprehensively has been introduced by KARAGIANNIS AND KÜHN (2002). Figure 8 illustrates the components of modeling methods and their relationships. By a first characterization, a modeling method can be divided into a *Modeling Technique* and *Mechanisms & Algorithms*. A Modeling Technique can be further partitioned into a *Modeling Language* and a *Modeling Procedure*. The concepts of Figure 8 are now described briefly:

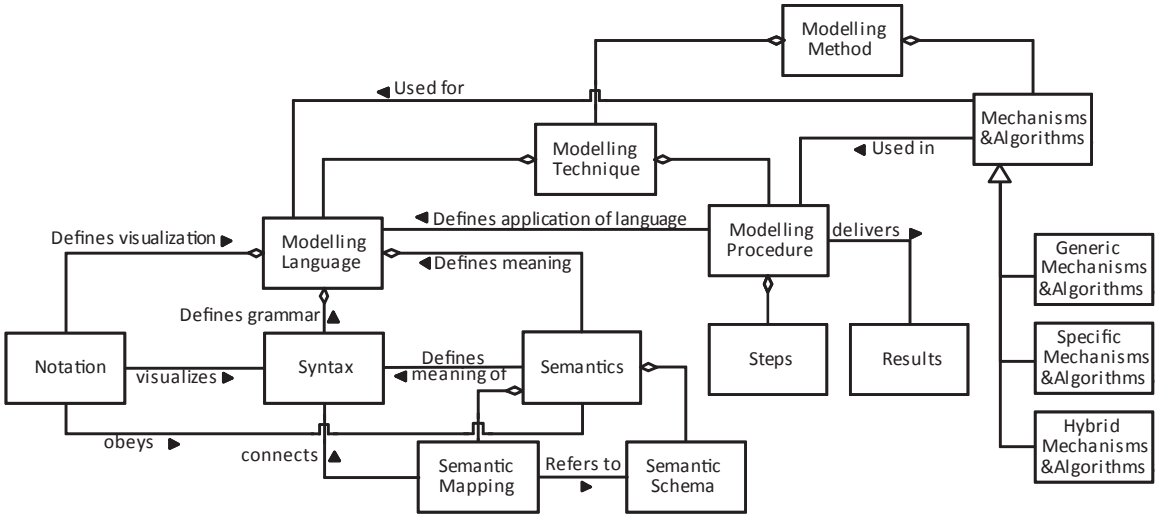


Figure 8: Components of modeling methods according to KARAGIANNIS AND KÜHN (2002)

- Modeling Language
  A modeling language, according to Karagiannis and Kühn, consists of (1) *Notation*, depicting the graphical representation of a modeling language e.g., by means of a shape, a figure, or a mathematically specified geometrical object for each of the modeling languages' elements; (2) *Syntax*, defining the grammar of the modeling language, i.e., specifying the modeling elements and the rules for connecting these elements with each other by means of a meta model (GEISLER ET AL., 1998), or a graph grammar (ROZENBERG, 1997) optionally supplemented by Object Constraint Language (OCL) OBJECT MANAGEMENT GROUP (OMG) (2010) rules for constraining valid models; and (3) *Semantics*, specifying the meaning of both, the modeling languages' elements defined in the syntax, and of the modeling language itself. Semantic specification is realized by a *Semantic Mapping*, referring the languages' elements to a *Semantic Schema*. The Semantic Schema may be realized formally by an ontology.

- Modeling Procedure
  A modeling procedure defines the *Steps* that need to be undertaken in order to build valid models according to a modeling language as well as the *Results* of applying the procedure. It therefore defines the application of a modeling language by executing *Mechanisms & Algorithms* of the method.

- Mechanisms & Algorithms
  Mechanisms & Algorithms are used to implement the functionality of the modeling language and the modeling procedure. Mechanisms are further divided into three categories (KÜHN, 2004, p. 26f.) : (1) *Generic Mechanisms*, (2) *Specific Mechanisms*, and

(3) *Hybrid Mechanisms*. (1) are implemented on the meta meta level, they can be used generally for any modeling method that is described as an instance of this meta meta model; (2) subsumes all mechanisms that are specific, i.e., they are built upon a certain meta model and therefore can only be applied to model instances of that meta model; (3) defines mechanisms that can be classified both, generic and specific, i.e., generic mechanisms on the meta meta level that are adapted or parameterized with concepts of a certain meta model on the meta model level.

Both presented frameworks are quite similar, however they differ in comprehensiveness and focus. The framework of Ferstl and Sinz aims at providing a general description of a modeling method, therefore comprising the most relevant aspects of a modeling method. The framework of Karagiannis and Kühn on the other hand aims at providing a comprehensive description schema with an additional emphasis on the implementation of a modeling method and model processing. Therefore comprising not only static and behavioral aspects of a modeling method but also mechanisms and algorithms, implementing the model processing functionality of a method. One central aspect of the framework by Ferstl and Sinz is the metaphor of a modeling method, whereas Karagiannis and Kühn considered the metaphor only inherently in their framework. Both frameworks are useful during the design and the documentation of modeling methods as they help managing the complexity of the method itself and therefore foster the inter-subjective understanding, applicability, and distribution of a modeling method. The framework of Karagiannis and Kühn has its strengths when it comes to the development of a modeling tool. A strength of the framework proposed by Ferstl and Sinz is the emphasis on the architecture model, i.e., aligning a procedure model to different models, sub-models, viewpoints, or perspectives of a method.

2.1.3 Meta Modeling

Meta models provide a formal specification of a modeling languages' syntax (HAREL AND RUMPE, 2004). (BÉZIVIN, 2005, p. 177) stated that *"A metamodel is a formal specification of an abstraction, usually consensual and normative"*. The specification of a meta model is also carried out using a modeling language, referred to as meta modeling language (cf. (KÜHN, 2004; FAVRE, 2005)). One step further, a model specifying the meta model is referred to as a meta meta model or meta^2 model. The number of meta levels is not fixed, however a hierarchy of four layers has been established: (Level 0) original, (Level 1) model, (Level 2) meta model, and (Level 3) meta meta model. This hierarchy is used in a number of popular meta modeling platforms, methods, and frameworks like Eclipse Modeling Framework (EMF)[4], MetaEdit+ (KELLY ET AL., 1996) , MetaObject Facility (MOF) (OBJECT MANAGEMENT

[4] `http://www.eclipse.org/modeling/emf/`, last checked: 2013-07-26

GROUP (OMG), 2011b), Architecture of Integrated Information Systems (ARIS) (SCHEER, 2000), Multi-Perspective Enterprise Modeling (MEMO) (FRANK, 2002), and Semantic Object Model (SOM) (FERSTL AND SINZ, 2013).

Using the layered approach breaks down the complexity and enables integration of the infinite number of possible models and meta models. Moreover, meta models, acting as a formalized specification of the modeling language, enable the integration of different modeling languages by relating the languages' meta concepts or model processing by tools. Level 0 is the original, i.e., aspects of the real world which are modeled in the model level (level 1). The model is defined using the concepts described in the meta model on level 2. On the most abstract level, level 3, the meta meta model defines the modeling language for creating meta models. Figure 9 visualizes the model hierarchy by highlighting the levels, languages, and the instantiation/classification relationships between the levels.

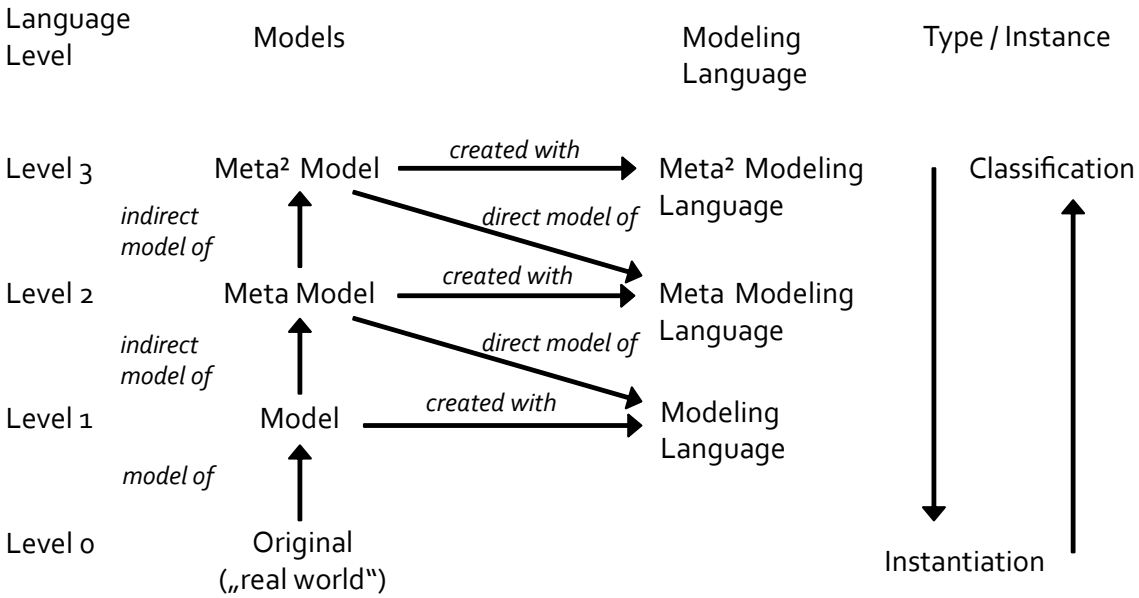


Figure 9: Meta modeling based on language levels (KARAGIANNIS AND KÜHN, 2002)

Although there is a common understanding of meta modeling, the language levels and a broad usage of meta modeling in industry and research, there is a differentiation according to the interpretation of the semantics according to the relationships between the meta levels. (FERSTL AND SINZ, 2013, p. 140f.) define an *extension* relationship between models on adjacent meta levels. The term extension is used, because the authors argue that for each concept on level n exists a higher number (or even an infinite number) of concepts derived by them on level n-1. Consequently, based on one meta meta model multiple meta models can be defined; using one meta model, multiple models can be defined; and one model can describe multiple real world situations.

KARAGIANNIS AND KÜHN (2002) on the other hand argue that there is an instantiation and classification relationship between models on adjacent meta levels. The instantiation relationship refers to the object-orientation programming paradigm, where classes define the abstract attribute types and abstract behavior. The instances itself set the concrete attribute values thereby realizing the concrete behavior. Every object is assigned a class it is instantiated of.

2.2 Multi-View Modeling

The complexity of today's world and the need to capture ever more aspects in models and relate them with each other requires modeling methods to adapt to this complexity by providing decomposition functionality. One approach of decomposing an overarching model, e.g., an enterprise architecture model comprising business process, organizational aspects, information technology, goals, information systems and machines, is to provide multiple viewpoints on this overarching model. The process of building a model is decomposed into several steps, each considering only the creation of a sub-area of the overarching model by utilizing a specifically designed modeling language, hereby creating a certain view on that model. The combination of the views gives the whole model. Notably, the relationship between viewpoint and view is considered analogous to the relationship between meta model and model. Hence, a view is considered an instance of a viewpoint, defined by means of a modeling language. This relationship is graphically visualized in Figure 10 with an emphasis on the semantic domain specified by a meta model and the subset of this semantic domain captured in the viewpoints and the accordingly created views.

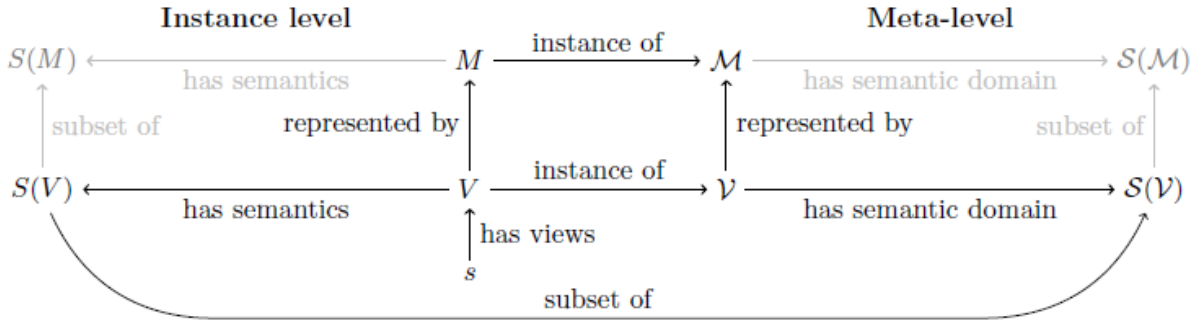


Figure 10: Relationships between views, viewpoints, models, and meta models (PERSSON ET AL., 2013)

The positive aspects of multi-view modeling, however, come at a price. Decomposing a coherent overarching model into multiple views inherently raises the need for dealing with the relationships and dependencies between those views. Hence, consistency between the views is vital for applicability and utility of multi-view modeling by human beings. PERSSON ET AL. (2013) identified a set of research challenges for multi-view systems:

Consistency Views should be consistent, i.e., the information covered in multiple views should not contradict each other.

Traceability It should be able to trace between elements of multiple views in order to gain an overview and to adhere to the inherent dependencies.

Reuse Content already captured in one view should be efficiently reused in other views, e.g., by referencing the content or by copying and pasting it.

Automation Tool development platforms should cover the specifics of multi-view modeling in order to provide tool developers with better development support when multi-viewing should be utilized.

Change propagation Changes performed in one view should be (semi-) automatically propagated to all other affected views.

Extendability The representation and the contents of a view should be extendable in an efficient way.

2.2.1 A brief history of Multi-View Modeling

Manifold approaches have been published that enable the analysis and specification of complex systems by reverting to multiple views. These approaches come from a wide range of domains. All of them specify their own semantics of viewpoints and views, however, they share the basic idea of decomposing a complex artifact into multiple views. This section introduces the most relevant approaches briefly, thereby highlighting strengths, weaknesses, and application scenarios. It will be shown, that the idea of multi-viewing is not new at all but still worth investigating.

2.2.1.1 Database Views

In the late 1970s, and early 1980s first scientific publications discuss the *View-Update Problem* (CARLSON AND ARORA, 1979; BANCILHON AND SPYRATOS, 1981; DAYAL AND BERNSTEIN, 1978, 1982; DAYAL AND HWANG, 1984). The databases grew in size and functionality, leading to problems of handling the large amount of data and dealing with inconsistencies. Providing views on a subset of the database entries was a way to handle the complexity. All views are based on the same database model. As the database views were not disjoint, changes performed on one view needed to be transformed into changes to the underlying database table; and possibly to changes to other database views. *"view operations are not performed directly but must be correctly translated into functionally equivalent operations on the database extension"* (CARLSON AND ARORA, 1979, p. 415). In that time, no automatism has been developed

for ensuring consistency of the database after view updates. A lot of research has been published that tried to handle this problem by describing conceptual and/or technical solutions. Figure 11 provides a formalized specification of the view-update-problem in relational databases[5].

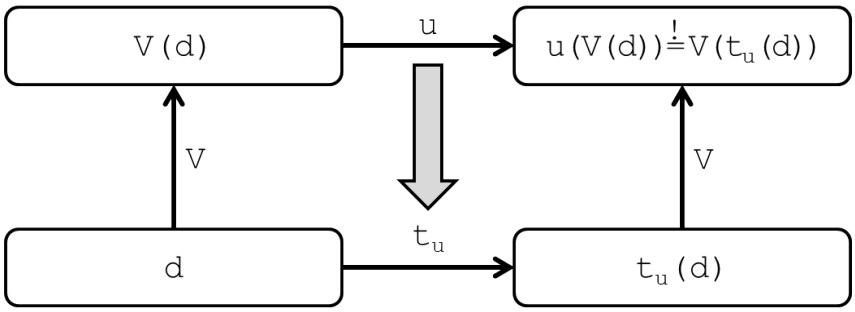


Figure 11: Formalization of the view-update-problem (VOSSEN, 1994, p. 188)

Figure 11 illustrates the consistency-preserving view-update mechanism for databases. A database schema d is consistent with a view $V(d)$, generated by a mapping function V applied to the current state of d. An update (u), performed on $V(d)$, generates a new state of the view, referred to as $u(V(d))$. The important challenge for a database management system is to transform the update u performed on the view into a semantically equivalent update t_u that needs to be applied on the database schema itself. Hence, transforming the database state from d to $t_u(d)$. Applying the mapping function V on the new database state results in $V(t_u(d))$. The view-update-problem is solved, if $u(V(d)) = V(t_u(d))$ while considering the integrity of the database and preventing side effects.

The *view update-problem* can be seen as one of the first approaches, suffering from the negative effects of having several views. However, compared to multi-view modeling, only dependencies in one direction must be considered as the views in a relational database schema are derived by a selection or projection operator, applied to one database. However, (DAYAL AND BERNSTEIN, 1982, p. 382) already stress, that a *"mapping is also required to translate view updates into corresponding updates on the conceptual schema extension. However, such an update mapping does not always exist, and even when it does exist, it may not be unique [...] So a change in the view extension may not be reflected unambiguously by equivalent changes in the database"*. (LECHTENBÖRGER, 2003, p. 49) also investigated in the view-update problem, he states, *"While read access through views is easy to handle, both from the users' as well as the system's perspective, update requests through views are difficult in the sense that they have to be translated into "appropriate" updates on the database relations". "In particular, the translations of updates should be handled transparently by the database system, i.e., be invisible to users, while the effects of translated updates should at the same time meet user expectations"*.

[5] This illustration is taken from lecture materials of the course database management systems of Prof. Sinz at the University of Bamberg

2.2.1.2 Requirements Specification Views

One of the pioneers in adopting multi-view modeling in the requirements specification domain was Anthony Finkelstein. Starting in 1989 (FINKELSTEIN AND FUKS, 1989; FINKELSTEIN ET AL., 1991, 1992), he published highly recommended papers on the utilization of multi-viewing towards a comprehensive requirements specification. Finkelstein and his colleagues not only specified multiple views on a conceptual basis, they also investigated on inconsistency handling (FINKELSTEIN ET AL., 1993) and the relationships between the multiple views (NUSEIBEH ET AL., 1994b). Finkelstein et al. proposed the *Viewpoints* framework for multi-view requirements specification which will be introduced briefly in the following.

The Viewpoints Framework

Finkelstein et al. introduced 1992 the Viewpoint Oriented Systems Engineering (VOSE) framework (FINKELSTEIN ET AL., 1992). The **multiple perspectives problem**, i.e., *"the problem of how to guide and organize development in this setting - many actors, sundry representation schemes, divers domain knowledge, differing development strategies"* (FINKELSTEIN ET AL., 1992, p. 33), served as motivation for the development of the VOSE framework. The central assumption of the framework was, that for the development of a composite system, several actors pursuing individual goals, thereby having a different knowledge base and using different representation styles are involved. Consequently, the requirements specification is realized by several *ViewPoints* on the system. A *ViewPoint "is a loosely coupled, locally managed object which encapsulates partial knowledge about the application domain, specified in a particular, suitable formal representation, and partial knowledge of the process of software development"* (KRAMER AND FINKELSTEIN, 1991, p. 247). Consequently, a complex setting is given thoroughly dominated by the overlaps of viewpoints, i.e., roles, responsibilities, knowledge.

A ViewPoint compromises the following constituents (referred to as *slots*) (FINKELSTEIN ET AL., 1992): a representation *style*, depicting scheme and notation of the information visualized by the ViewPoint (e.g., a modeling language); a *domain*, representing the sub area of the real world relevant for the representation; a *specification*, i.e., the resulting statements depicted in the representation (e.g., the models created according to the modeling language); a *work plan*, describing the sequence of tasks performed in order to build a specification (i.e., *Assembly Actions*), to create new ViewPoints (i.e., *ViewPoint Trigger Actions*), and to check consistency within and between overlapping ViewPoints (i.e., In- and Inter-ViewPoint Check Actions); a *work record*, realizing a history of the specification documents by referencing the tasks of the work plan to their completion status.

The VOSE framework can be applied to create a comprehensive requirements specification of a system. The framework has an emphasis on the users, agents, and roles that are partic-

ipating in the design of the system. Each viewpoint utilizes a specific representation style, most suitable for its intended usage. Consistency issues, referred to as *"in-viewpoint checks"*, and *"inter-viewpoint checks"* raised by overlapping viewpoints, are handled in an early stage by introducing consistency checks in the work plan. However, the consistency management is only on an informal level. The quality of the comprehensive specification, i.e., the number of considered and handled inconsistencies, relies strongly on the people defining the viewpoints. Subsequent publications have tried to discuss this problem, however, the authors argue, that *"maintaining consistency in multipespective software development is not always possible"* (NUSEIBEH ET AL., 1994b,a; EASTERBROOK ET AL., 1994).

2.2.1.3 Information System Development Views

First publications of *Multiview1* have been made by Wood-Harper et al. in 1985 (WOOD-HARPER ET AL., 1985) with some conceptual foundations published in 1982 (WOOD-HARPER, 1982). Subsequently, the method[6] has been used and further developed (AVISON AND WOOD-HARPER, 1990; WOOD-HARPER AND AVISON, 1992), leading to the publication of Multiview2 in 1998 (AVISON ET AL., 1998).

The Multiview framework was directed towards Information Systems (IS) development *"that would take account of the complex world of people and organizations as well as information and communication technologies"* (AVISON ET AL., 1998, p. 124). Based on the analysis of existing, i.e., conventional and structured methods, the authors defined three major shortcomings the Multiview framework should resolve (AVISON ET AL., 1998): (1) *The narrow scope of conventional approaches*, the stakeholders and the aims of the organization using information systems are not considered appropriately in the conventional approaches; (2) *The rigidity in use of conventional approaches*, most conventional approaches are designed according to a *"ideal type"* and provide solutions limited to that ideal type - which is not a common thing in reality; and (3) *Adherence to the waterfall model*, the commonly known top-down and step by step approaches are not suitable for scenarios in reality as users do not always behave like they should and switch or skip steps.

The goal of Multiview1 was therefore to provide an alternative, more flexible IS development framework that brings together the computer specialists implementing the IS, and the users who are designated to use it. The authors defined five stages in the application of the framework, each directed to answer a specific question (AVISON AND WOOD-HARPER, 1991): (1) analysis of human activity (*"Q1: How is the information System supposed to further the aims of the organization using it?"*); (2) analysis of information (*"Q2: How can it be fitted into the work-*

[6] The authors define a Multiview methodology. However, the definition of the terms 'method' and 'methodology' is not precise and unambiguously up to now. Therefore, the term 'methodology' is replaced by 'method' consistently in this thesis. The interested reader is referred to OLIGA (1988) for a distinction of the terms.

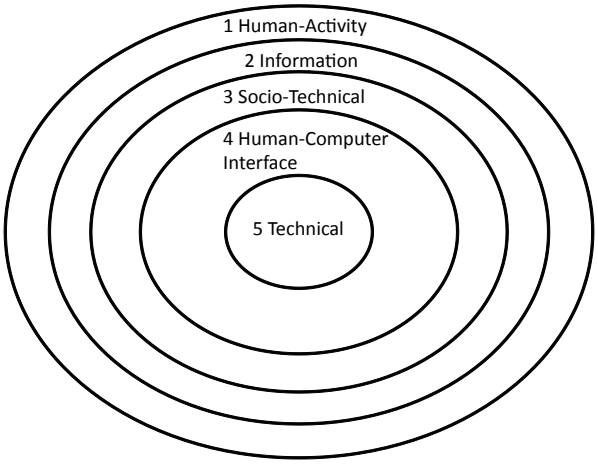


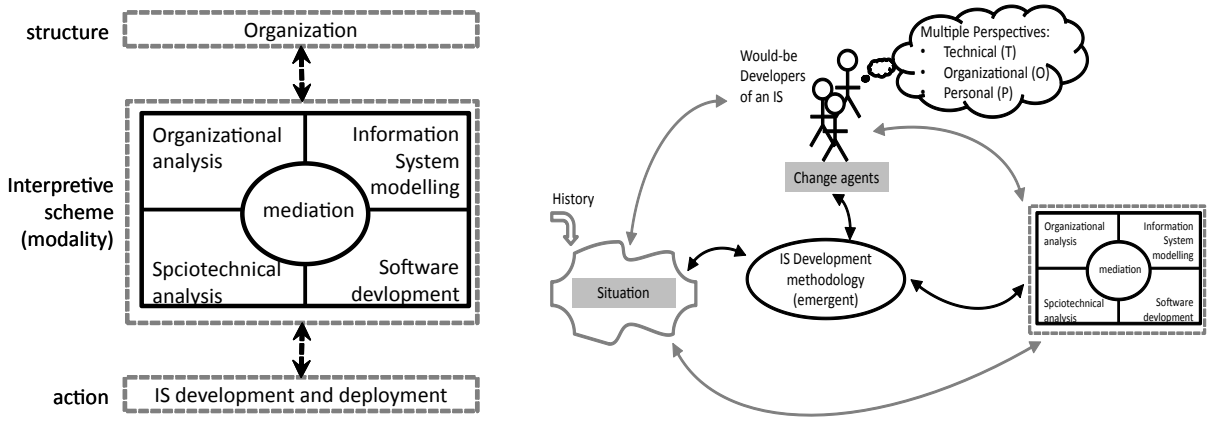
Figure 12: The Multiview1 framework (AVISON AND WOOD-HARPER, 1991)

ing lives of the people in the organization using it?"); (3) analysis and design of socio-technical aspects (*"Q3: How can the individuals concerned best relate to the computer in terms of operating it and using the output from it?"*); (4) design of the human-computer interface (*"Q4: What information processing function is the system to perform?"*); (5) design of technical aspects (*"Q5: What is the technical specification of a system that will come close enough to meeting the identified requirements?"*). Each of these five stages defines a different view on the information system.

Figure 12 illustrates the five stages of the Multiview1 method. The circles visualize a decreasing specificity and an increasing human factor from the center (*5 Technical*) to the margin (*1 Human-Activity*), whereas in the opposite direction, there is an increasing specificity *"and the progressive development of an information system"* (AVISON ET AL., 1998, p. 127).

As mentioned earlier, the method has been applied and evaluated in a number of case studies (AVISON AND WOOD-HARPER, 1990, 1991). In 1998, the experience and information gained through these applications led to the evolution of *Multiview2* (AVISON ET AL., 1998). Whereas Multiveiw1 has not included the steps of software development, implementation, and production operation, Multiview2 does. The framework has been widened, in order to incorporate some aspects of information system development.

Figure 13 (a) visualizes the *interpretive scheme* which is positioned between the organizational structure, the developed information system is to be embedded in, and the actual development of the information system, possibly having an effect on the organizational structure. Aspects on the organization, work, and technicality of Multiview1 have been preserved, but extended with development-specific ones (i.e., *"building, implementing, and maintaining an information system"* (AVISON ET AL., 1998, p. 130)). The process of information systems development plays a mediation role (see Figure 13 (a)). Mediation means, that all views are



(a) Multiview2 structure

(b) Multiview2 elaborated framework

Figure 13: The Multiview2 framework (AVISON ET AL., 1998)

present at all time and that the IS development process should not exclusively be seen as a way to bridge the domains of human beings and technique. Figure 13 (b) visualizes the interplay of the components of Multiview2 by highlighting the integration of Linstone's multiple perspective concept (LINSTONE, 1984) and the emergent process application. A detailed specification of all five stages (i.e., views) including the decomposition into respective *"substages"*, the specific methods utilized, the goals pursuit, and the tasks performed, can be found in AVISON AND WOOD-HARPER (1990).

2.2.1.4 Enterprise Modeling Views

Because of the complexity of modern enterprises, modeling of such enterprises is performed since decades using multiple specifications (referred to as views, viewpoints or perspectives in the relevant literature). In the following, one older approach (Multiple Perspective Concept), one commonly used approach (Architecture of Integrated Information Systems (ARIS)) and current enterprise modeling approaches are briefly discussed. The research in and the development of these methods considerably contributed to the theoretical and conceptual foundations of multi-view modeling, as we refer to it nowadays.

Multiple Perspective Concept

The *Multiple Perspective Concept* introduced by LINSTONE (1984) comprises three perspectives: (T) a technical perspective; (O) an organizational or societal perspective, and (P) a personal or individual perspective. All three perspectives realize a certain view on the system by applying a "filter" (LINSTONE, 1989, p. 312). Linstone justifies the need for multiple perspectives as follows: *"(1) each perspective yields insights not obtainable with the others; and (2)*

the O and P perspectives are essential in bridging the gap between analysis and action“ (LINSTONE, 1989, p. 314). In the following, the three perspectives are introduced briefly.

The *technical perspective* takes a science-technology view on the reality with a far planning horizon. The perspective is directed towards problem solving by applying abstraction, idealization, and isolation (LINSTONE, 2003). The mode of inquiry for this perspective is by observation, analysis, data, and models.

The *organizational perspective* utilizes an unique group or institutional view on the reality considering an intermediate planning horizon. The focus of this perspective lies on the processes, actions, and roles in an enterprise. Central questions answered with this perspective, are *“does something need to be done, and if so, what?“ and “who needs to do it and how?“* (LINSTONE, 2003, p. 5)

The *personal perspective* takes the perspective of a certain human being (an unique individual) on the reality, considering a short (for most) planning horizon. This perspective reflects all aspects that are not considered in T and O perspective. As the perspective is a personal one, it depends strongly on the individual and its role. Intuition, learning, and experience are utilized for inquiry.

ARIS

The ARIS framework, developed by SCHEER (2000), can be used to create a comprehensive description of the information system architecture of an enterprise. ARIS defines five different views to describe an enterprise: *the function view, the organization view, the data view, the output view, and control view/process view* (SCHEER AND SCHNEIDER, 2006, p. 609). These multiple views are conceptually dependent, however their integration is not specified, hence, needs to be performed cognitively by the modeler. *“While ARIS allows for various perspectives on the enterprise [...], the integration of these aspects is somewhat lacking“* (LANKHORST, 2009, p. 39).

Current Enterprise Modeling Approaches

The list of enterprise modeling methods utilizing multi-viewing could be easily extended e.g., by Computer Integrated Manufacturing Open System Architecture (CIMOSA) defined by AMICE Consortium for an ESPRIT (European Strategic Program for Research and Development in Information Technology) project ESPRIT CONSORTIUM AMICE (1998), Multi-Perspective Enterprise Modeling (MEMO) defined by FRANK (2002), Integrated DEFinition Methods (IDEF) (MENZEL AND MAYER, 2006), defined by the U.S. Air Force Integrated Computer Aided Man-

ufacturing program[7], the Semantic Object Model (SOM) defined by FERSTL AND SINZ (2013). The interested reader is referred to section 4, where most of the denoted enterprise modeling methods are analyzed.

2.2.1.5 Software Architecture Views

Multi-viewing is extensively applied in the domain of software architectures. With the increasing complexity of software systems and the increasing need to integrate software components with its environment, modeling methods for the specification of these architectures need to provide capabilities to handle complexity. Hence, multi-viewing is supposed to be capable of breaking down the complexity by decomposing the overarching software architecture into several viewpoints. In this domain, viewpoints are considered to be strictly aligned to and designed for the specific needs of a (group of) stakeholder(s). Hence, a viewpoint is specified by the set of concerns it addresses. These concerns were derived from the needs of the stakeholder(s).

Kruchten's 4+1 View Model

First ideas of approaching multi-viewing in the software architecture domain have been proposed in 1995 by Philippe Kruchten (KRUCHTEN, 1995). He introduced the $4+1$ view[8] model for software architectures. A model, used *"for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views"* (KRUCHTEN, 1995, p. 42). An architecture model, according to KRUCHTEN (1995) consists of five views: a *logical view* realized as a object model in an object-oriented design method (i.e., classes and connectors between classes as central model components); a *process view* realized as process model (i.e., processes, messages and events as central model components); a *physical view* considering especially non-functional requirements of the software system (i.e., processor, devices, and communication connectors as central model elements); a *development view*[9], realized as a architecture model hierarchically decomposing the software to modules, subsystems and layers; and a *scenario view* summarizing and organizing the definitions of the other four views, hereby concentrating on the interplay of the different views.

Figure 14 illustrates the relations between the five views of the architecture model. The views are not independent from each other: e.g., classes from the logical view are mapped onto tasks and processes in the corresponding process view; classes in the logical view are related to modules and packages in the development view; processes or process groups of the process

[7] The IDEF family of methods specifications can be found at: `http://idef.com/`, last checked: 2015-02-28

[8] Although not in line with the understanding of views and viewpoints of this thesis, the term view is used consistently as introduced by Kruchten KRUCHTEN (1995) in the following.

[9] later referred to as *implementation view* KRUCHTEN (1996, 2000)

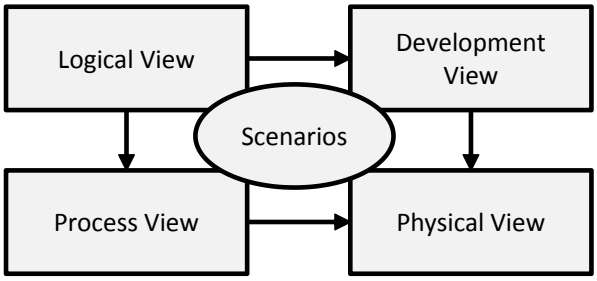


Figure 14: The "4+1" view model (KRUCHTEN, 1995)

models are mapped onto the available physical hardware in the physical view; and many more (cf. KRUCHTEN (1995) for a full list of view relations).

ISO/IEC/IEEE 42010

The benefits of approaching software architectures using multiple views even led to an industry standard. The first architecture standard was released in July 2007 (cf. ISO/IEC 42010:2007) by adopting the former IEEE 1471 standard to a cooperative international standard of IEEE and ISO (cf. EMERY AND HILLIARD (2008)). In 2011, the current version of the standard has been released, *ISO/IEC/IEEE 42010:2011* (IEEE, 2011). Originally, the standard was dedicated for defining the requirements on architectures of software-intense systems. In the latest version, the aim is also reflecting *"creation, analysis and sustainment of architectures of systems through the use of architecture descriptions"* (IEEE, 2011, p. v). Moreover, 42010:2011 *"can be used to establish a coherent practice for developing architecture descriptions, architecture frameworks and architecture description languages within the context of a life cycle and its processes"* (IEEE, 2011, p. v).

In the following, the conceptual model of the architecture description of IEEE 42010:2011 is discussed briefly (IEEE, 2011; HILLIARD, 2012, p. 2ff.)[10]:

- **System of Interest**: the system whose architecture is under consideration in the preparation of an architecture description.
- **Architecture (System)**: fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.
- **Architecture Description**: work product used to express an architecture. An architecture description is composed of one or more architecture views, expressing the architecture of the system of interest in accordance with an architecture viewpoint.

[10] The following definition of the concepts is strongly aligned to the definition in the international standard. Having an international standard obviates an interpretation of the concepts definition.

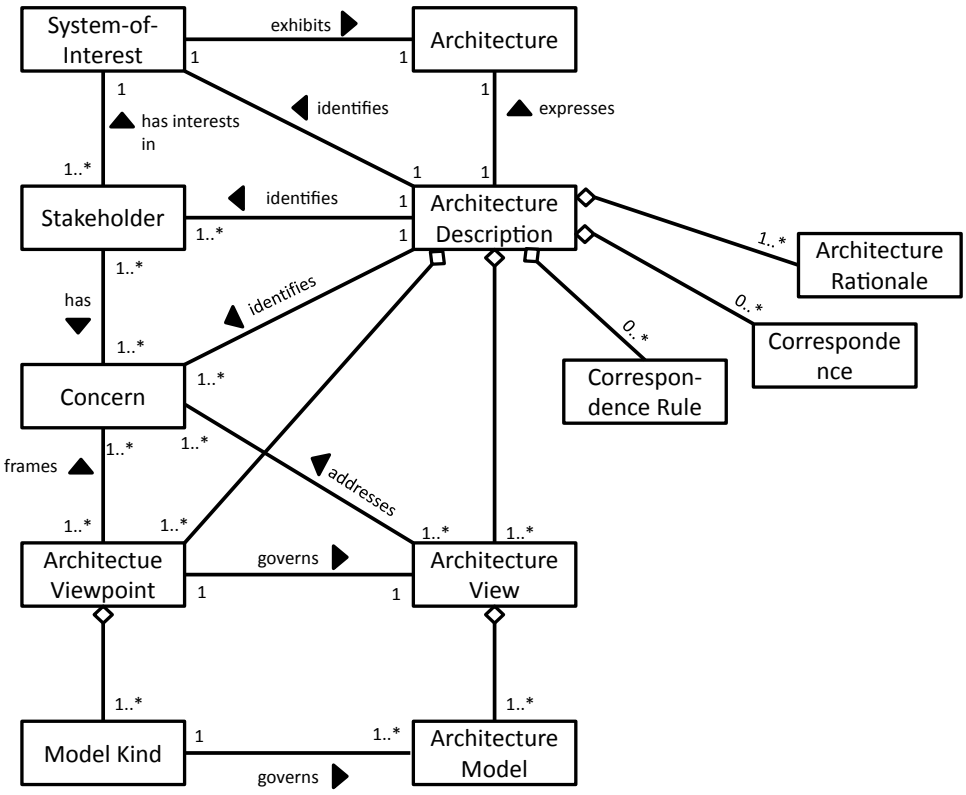


Figure 15: Conceptual model for an architecture description (IEEE, 2011, p. 5)

- **Stakeholder (System)**: individual, team, organization, or classes thereof having an interest in a system. Each stakeholder of the system of interest usually has one or more concerns about the system.
- **Concern (System)**: interest in a system relevant to one or more of its stakeholders. A concern could be manifest in many forms, such as in relation to one or more stakeholder needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues pertaining to the system.
- **Architecture Viewpoint**: work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns. Viewpoints specify two aspects: (1) the concerns they frame for stakeholders, and (2) the conventions they establish for architecture views.
- **Architecture View**: work product expressing the architecture of a system from the perspective of specific system concerns. Architecture views are generated by relying on the conventions established by the corresponding architecture viewpoint. Each architecture

view is composed of one or more architecture models.

- **Model Kind**: conventions for a type of modeling (e.g., languages, notations, modeling techniques, analytical methods and other operations (HILLIARD, 2012, p. 5). The model kind governs the specification of architecture models.
- **Architecture Model**: a work; it's subject is determined by its model kind. An architecture model uses modeling conventions appropriate to the concerns to be addressed. These concerns are specified by the model kind governing that model.
- **Architecture Rationale**: records explanation, justification or reasoning about architecture decisions that have been made. The rationale for a decision can include the basis for a decision, alternatives and trade-offs considered, potential consequences of the decision and citations to sources of additional information.
- **Correspondence**: defines a relation between architecture description elements. Correspondences are used to express architecture relations of interest within an architecture description (or between architecture descriptions).
- **Correspondence Rule**: used to enforce relations within an architecture description (or between architecture descriptions). Correspondence rules govern correspondences by expressing and enforcing architecture relations such as composition, refinement, consistency, traceability, dependency, constraint, and obligation.

Several authors used the conceptual model of the IEEE to define their own conceptual model for multi-view architecture descriptions. The most relevant will be discussed briefly. It has been introduced by Dijkman (DIJKMAN ET AL., 2006; DIJKMAN, 2006; DIJKMAN ET AL., 2008). He proposed a more generic evolution of the IEEE architecture model (cf. Figure 15) with an emphasis on the relationships between the views and the modeling languages utilized to create the views. The elements of multi-viewpoint design according to DIJKMAN ET AL. (2006) are visualized in Figure 16.

2.2.1.6 Software Modeling Views

Atkinson (ATKINSON ET AL., 2010, 2013a,b) 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 approach is built with UML models targeting at the model-driven development of software systems. Hence, the focus is on how to specify, and later on transform, software systems based on multiple views. *"The key idea behind SUM-based software engineering environments is to separate concerns - the SUM is responsible for*

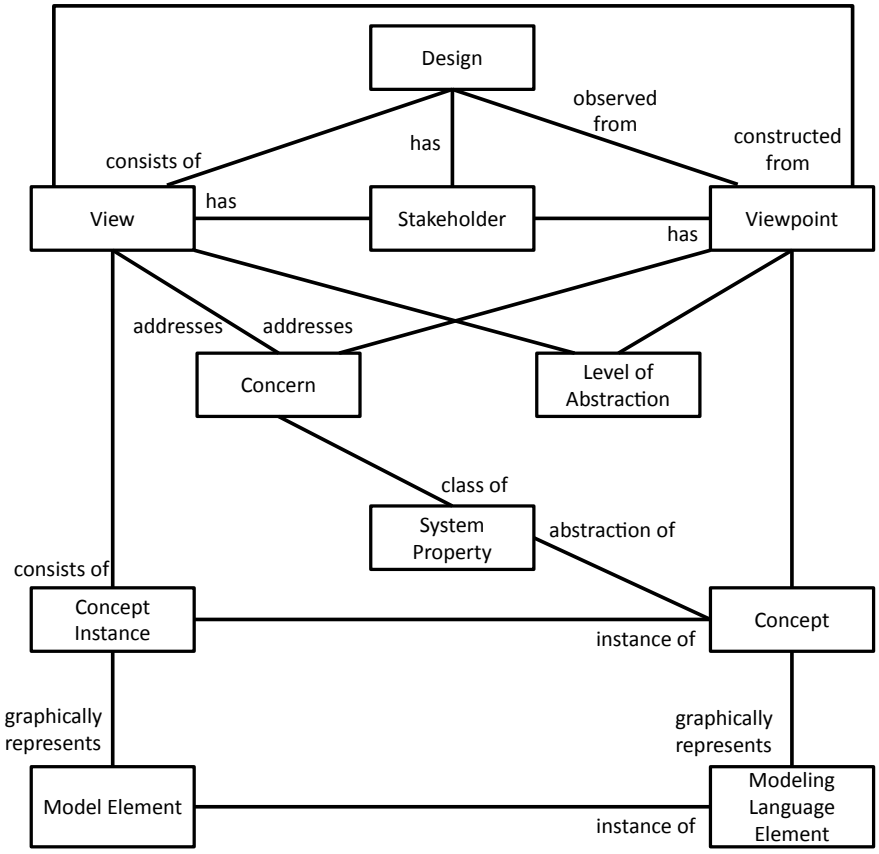


Figure 16: Multi-Viewpoint Design (DIJKMAN ET AL., 2006)

storing all known information about the system at the optimal level of abstraction, with minimal redundancy, while the views are responsible for presenting and/or editing carefully chosen projections of this information" (ATKINSON ET AL., 2013a, p. 2:6).

Recent publications of KRAMER ET AL. (2013) differentiate themselves from OSM by providing flexible, not predefined viewpoints for modeling software systems. An emphasis is also on the correspondences between the viewpoints and mechanisms for ensuring consistency. Like for OSM, the focus is on using the multiple UML views for software development purposes.

2.2.1.7 Software Engineering Views

"Contemporary software systems combine many artifacts specified in various modeling and programming languages, domain-specific and general purpose as well. Since multi-language systems are so widespread, working on them calls for tools with cross-language support mechanisms such as (1) visualization, (2) static checking, (3) navigation, and (4) refactoring of cross-language relations" (PFEIFFER AND WASOWSKI, 2012a,b, 2015).

One of the first publications that combined multiple viewpoints in requirements analysis and

software development has been written by ROSS AND SCHOMAN (1977). In 1977, the authors already identified the huge number of stakeholders (they called it roles) in the process of defining requirements and transforming them into software systems. The method, developed to decompose the system specification according to the needs of multiple stakeholders into views, was called Structured Analysis and Design Technique (SADT). The authors propose, *"Each subject - context analysis, functional specification, and design constraints - must be examined from at least three points of view: technical, operational, and economic"* (ROSS AND SCHOMAN, 1977, p. 10).

The *GOOSE* method, developed by REEVES ET AL. (1995), was aimed to create a formalized *design model* of a software artifact. This design model was decomposed into four viewpoints: *functional viewpoint, behavioral viewpoint, structural viewpoint, and data modeling viewpoint*. The authors defined the viewpoints as projections on the comprehensive design model. The motivation for GOOSE was in the high number of unsuccessful software development projects. Hence, the design model was aimed to capture requirements on a more formalized and abstract level, bridging the gap between different stakeholders, e.g., designer and developer, or requirements engineer and developer.

In 1996, Soni et al. emphasized on the usefulness of specifically described multiple views in software development by the following hypothesis: *"...quality and productivity in software development would be improved if software designers were to arrive at similar agreements. In other words, we need to view a software system from a variety of perspectives, precisely describe these perspectives, separate these descriptions to manage complexity, establish correspondence, and facilitate the analysis and manipulation of these descriptions. We do not want to imply that informal descriptions of design are useless, however, we do believe that their usefulness is rather limited"* (SONI ET AL., 1996, p. 6). Notably, Soni et al. distinguish between precisely and informal descriptions of the perspectives. The authors propose the usage of five perspectives during software development: *source code, code architecture, module interconnection architecture, execution architecture, and conceptual architecture*.

2.2.1.8 Summary

In the preceding sections, several ways of adopting multi-view modeling in manifold domains have been presented. The goal was not to describe every approach comprehensively and to compare each and every one with the MUVIEMOT method. The goal was to establish a sensibility on the importance, reputation and origins of multi-view modeling as it is often misunderstood as a pure visualization means. All of these approaches have their own understanding of what constitutes a view, how views are created and edited, and how consistency is preserved. Most of the approaches are on a theoretical and conceptual, hence, informal level. The thesis at hand tries to fill that gap by introducing a formalized foundation for multi-view modeling in the following.

As has been mentioned already in the introduction of this section, the description does not claim to be comprehensive. Multi-view modeling, in many different incarnations, has been applied in several other domains and kinds, the interested reader is referred to e.g., Mega Modeling (BÉZIVIN ET AL., 2003; BARBERO ET AL., 2007; FAVRE, 2004; HEBIG ET AL., 2011), Multi-Formalism Modeling (VITTORINI ET AL., 2004; SANDERS, 1999), Multi-Modeling (YIE ET AL., 2009), or domain-specific meta-modeling (DE LARA ET AL., 2013).

2.2.2 View, Viewpoint & Multi-View Modeling Definitions

In order to establish a common understanding of the foundations of multi-view modeling, it is inevitable to define the concepts **view**, **viewpoint**, and **multi-view modeling** precisely. But first, some definitions found in the divers domains that utilize multi-view modeling are presented. A common understanding of the terminology, followed by in this thesis, will be given in section 5.

The *IEEE standard 42010* defines a view as *"a representation of a system from the perspective of a related set of concerns"* and the corresponding viewpoint as *"a specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis"* (IEEE, 2011).

In the context of Kruchten's work on software architectures (KRUCHTEN, 1995) and the Reference Model for Open Distributed Processing (RM-ODP) (UNION, 1997, p. 9), the following definition can be found: *"A viewpoint (on a system) is an abstraction that yields a specification of the whole system related to a particular set of concerns"*. Staying in the architecture domain, Radjenovic and Paige used the following definition: *"A view is a representation of a whole system, or its parts, from the perspective of a related set of concerns"* (RADJENOVIC AND PAIGE, 2008, p. 202), as an extension of the IEEE definition introduced above. Dijkman followed this direction by defining a view as follows: *"The viewpoint of a stakeholder is the combination of the concerns and levels of abstraction that the stakeholder addresses and the set of concepts that the stakeholder uses to construct his or her part of an overall design"* (DIJKMAN, 2006, p. 21).

A more pragmatic and domain-independent definition comes from (BROY, 2003, p. 210): *"A view is - like a model - an abstraction that concentrates on a particular aspect of a software system"*. Even more simplified is the definition by Lankhorst: *"Simply put, a view is what you see, and a viewpoint describes from where you are looking"*(LANKHORST, 2009, p. 154). Aspects concerning the creation, manipulation and analysis of views have been introduced with this definition: *"A viewpoint is a specification of the conventions for using a view, by establishing the purposes and audience for a view, and the techniques for its creation and analysis"* (PERSSON ET AL., 2013, p. 1).

According to (VON HANXLEDEN ET AL., 2012, p. 1), *"Multi-view modeling refers to a*

system designer constructing distinct and separate models of the same system to model different (semantic) aspects of a system". (REINEKE AND TRIPAKIS, 2014, p. 1) propose a more plain definition: *"Multi-view modeling is a methodology where different aspects of the system are captured by different models, or views"*. Brooks et al. define multi-view modeling as *"[...] distinct and separate models of the same system are constructed to model different aspects of the system"* (BROOKS ET AL., 2008, p. 1). SHAH ET AL. (2010) describe multi-view modeling as the process *"in which a system is described from multiple points of view"* (SHAH ET AL., 2010, p. 580).

The definitions do not aim to be comprehensive - there are numerous more, slightly differing definitions in the relevant literature. By contrast, the aim was to show the diversity of understandings of the fundamental concepts of this thesis.

2.2.3 Viewpoint Relationships

In multi-view modeling, the viewpoints are normally not independent from each other. More precisely, the viewpoints are somehow related, not only on a syntactic but also on a semantic basis. Syntactic relationships exist, if modeling concepts are part of several viewpoints. These relationships can be automatically detected by looking at the overlapping areas of the respective meta models or by identifying shared concepts of a given common meta model. By contrast, semantic relationships are not identified that easy. To reveal these relationships, one needs to have in depth knowledge of the semantics of the different viewpoints. In the following, an overview of the most related literature on viewpoint relationship types in multi-view modeling is given (see e.g., (BOUCKÉ ET AL., 2008; RADJENOVIC AND PAIGE, 2008; WEIDLICH ET AL., 2010) for detailed analysis of relationship types).

PERSSON ET AL. (2013) provide a comprehensive survey of relationships in integrated multi-view modeling of cyber-physical systems. The authors analyze different relationship types using three categories: *content relationships*, *process relationships*, and *operation relationships*.

Content Relationships Persson et al. analyze different overlap types considering the modeling language of viewpoints. They distinguish between *semantic* and *syntactic* overlaps. The former is visualized in Figure 17 with a circle and the symbol 'S' whereas the latter is visualized with a circle and the symbol 'M' (for modeling language). As the illustration shows, three types of overlaps are distinguished: (1) *no overlap*, (2) *partial overlap*, and (3) *complete overlap*, i.e., containment. In the syntax category, only the first two types appear. If two views are syntactically overlapping, they must share as well semantics. Vice versa, a semantically overlapping area does not imply a syntactic overlap.

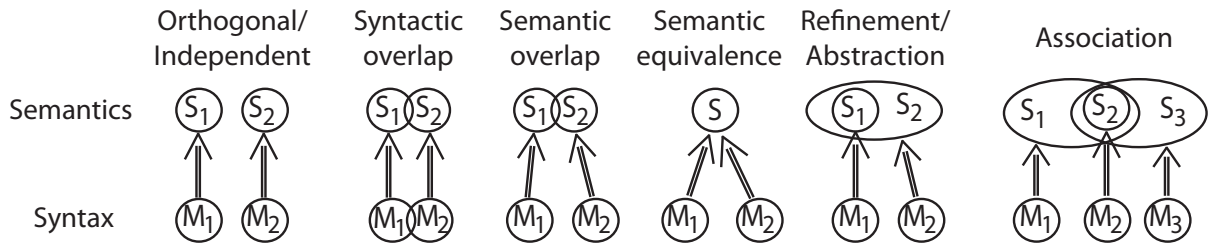


Figure 17: Classification of viewpoint content relationships (PERSSON ET AL., 2013)

Process Relationships Process relationships *"are based on how the views relate to the developmental workflow"* (PERSSON ET AL., 2013, p. 5). This category investigates three relationship types: (1) *precedence relationship*, a certain view has to exist before another view without sharing data; (2) *dependency relationship*, a certain view has to exist before another view - with the second view containing data defined in the first view; and (3) *co-dependency relationship*, circular shared data between two views.

Operation Relationships The *operations category* investigates the manipulation, i.e., creation, or combination of views. The authors distinguish six classes of operation types: *composition*, *projection*, *extension*, *analysis*, *synthesis*, and *general case* (see (PERSSON ET AL., 2013, p. 5f.) for a comprehensive survey).

Due to the nature of multi-view modeling methods, identification, specification, and management of relationships between viewpoints is vital for the efficient application of the methods. Moreover, these relationships are necessary for the design of consistency-preserving modeling tools. From a tool developer's perspective, content relationships need to be transformed into accordingly implemented consistency functionality; process and operation relationships need to be implemented by means of a tools' modeling procedure.

LOCHMANN AND HESSELLUND (2009) describe three types of content relationships they identified in multiple domain-specific modeling language settings. Figure 18 illustrates the three categories: *Explicit Typed Reference*, references that are described by relating types (i.e., modeling concepts) of respective meta models to each other; *Implicit Reference*, references that are based on properties of the created models, i.e., two modeled elements in different models have the same name attribute value; and *Complex Semantic Connections*, semantically further constraining the references, i.e., not just referential integrity between two modeling concepts.

The referenced works document the need for a thorough investigation of the behavioral aspects of multi-view modeling, i.e., how the modeler interacts with multiple views, how changes, performed on one view are transformed into changes that need to be applied to other views. For a tool developer, it is of major importance, that these relationships are specified in a formalized way and at the meta model level. This eases the transformation of the dependencies into functionality of a modeling tool.

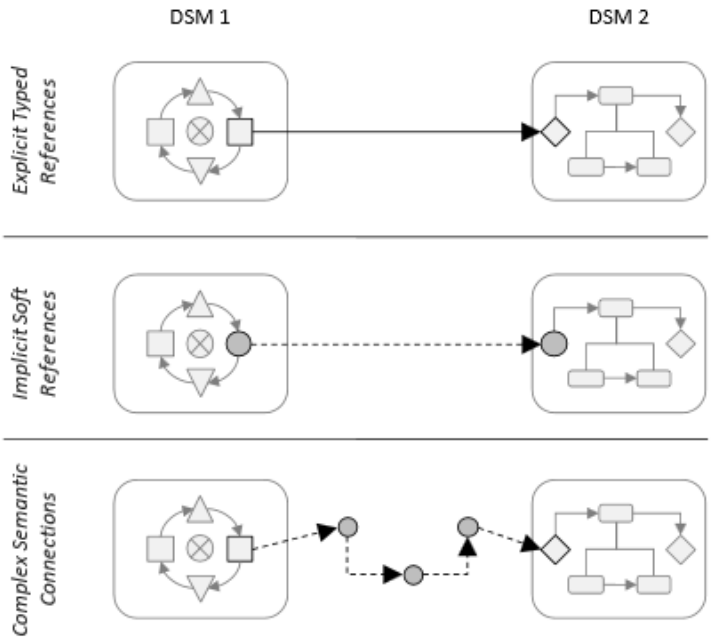


Figure 18: Explicit, implicit and complex semantic viewpoint relationship types (LOCHMANN AND HESSELLUND, 2009)

2.2.4 Consistency

An important prerequisite of successful applications of multi-viewing is consistency between the multiple views. Lopez-Herrejon et al. state that a *"crucial demand of MVM systems is consistency checking to describe and preserve the semantic relationships amongst the elements of the different views"* (LOPEZ-HERREJON AND EGYED, 2011, p. 348). In order to establish consistency, one needs to be aware of the syntactic and semantic relationships between the views (cf. section 2.2.3). Consistency management is vital for the utility of multi-view modeling approaches, independent from the application domain. Hence, it is no surprise, that lots of research has been directed towards consistency management (e.g., (HUZAR ET AL., 2005; MENS ET AL., 2005; USMAN ET AL., 2008; BHAVE ET AL., 2011; HERZIG ET AL., 2011; SHAH ET AL., 2010; LOPEZ-HERREJON AND EGYED, 2011)).

Before thinking about consistency management, the term consistency itself needs to be specified in the context of multi-view modeling. The definition of conformity is first introduced to guide the way until an understanding of the more stringent definition of consistency. Larkin and Simon differentiate *informational and computational equivalency* (LARKIN AND SIMON, 1987). Informational equivalence: *"two representations are informationally equivalent if all information in one is also inferable from the other and vice versa."*. Computational equivalence: *"two representations are computationally equivalent if they are informationally equivalent and,*

in addition, any inference that can be drawn easily and quickly from the information given explicitly in the one can also be drawn easily and quickly from the information given explicitly in the other, and vice versa.“.

HERZIG ET AL. (2011) differentiate between *internal and external inconsistency*. The former considers a syntactic conformity of the created view to its syntactic specification, i.e., the viewpoint, whereas the latter furthermore considers whether the mapping of the reality to the constituents of the view is correct. Later in 2014, Herzig et al. proposed an approach for managing inconsistencies based on the hypothesis, that all models can be represented by graphs and that inconsistencies can be identified by means of pattern matching techniques (HERZIG ET AL., 2014).

A quite simple specification of consistency is given by Reineke et al. who state that views should *“not contradict each other“* (REINEKE AND TRIPAKIS, 2014, p. 1). In a similar direction goes the definition of the term consistency in the Oxford Dictionary[11] defining consistent for an argument or set of ideas as *“not containing any logical contradictions“*.

Multi-view consistency is classified in several ways in literature. Some of them are as follows:

Weak and strong consistency BHAVE ET AL. **(2011)** This is related to the scale of connection between the models of a system. *Weak consistency* requires every architectural artifact, i.e., components and connectors, of a system and every connection between different views to be considered in a model. On the other hand *strong consistency* also adds a constraint that every view should consider all the artifacts, i.e., components and connections.

Vertical and horizontal consistency BROY ET AL. **(2010)** *Vertical consistency* refers to consistency between models at different development phases or abstraction levels. *Horizontal consistency* on the other hand refers to the consistency between models at the same phase, representing different views.

Global and local consistency QURESHI **(2012)** *Global consistency* is considered from a set of global constraints and relationships. A *locally consistent* model can refer to a model whose artifacts are consistent with each other.

Syntactic and semantic consistency *Syntactic consistency* considers the conformity of a view to its abstract syntactic specification, i.e., its viewpoint. *Semantic consistency* is concerned with the semantic equivalence of aspects captured in multiple views (cf. ENGELS ET AL. (2001) for syntactic/semantic consistencies in UML).

[11] The Oxford Dictionary page for the term consistent: `http://www.oxforddictionaries.com/definition/english/consistent`, last checked: 2015-03-01

As the previous literature review suggests, there are manifold approaches trying to classify consistency. Notably, approaches that explicitly require or at least tolerate inconsistencies (e.g., (BALZER, 1991; NUSEIBEH ET AL., 2001)) have not been regarded, as the preserving of consistency in multi-view modeling is considered vital. In the following, a solid excerpt of the most relevant consistency classes is introduced in detail. These classes will be used throughout the thesis.

Multi-View Modeling Consistency

A lot of publications around the Unified Modeling Language (UML) deal with the problem of inconsistencies in a multi-view context (ENGELS ET AL., 2001; LOPEZ-HERREJON AND EGYED, 2011; MENS ET AL., 2005; USMAN ET AL., 2008)[12]. This is no surprise as the UML, besides all positive aspects, lacks at formalization (HAREL AND RUMPE, 2004; SEIDEWITZ, 2013). Authors discuss inconsistency issues faced while trying to combine multiple UML diagrams. However, on an abstract level, those consistency issues, and the abstract consistency classes derived from them, are also applicable for non-UML models (cf. BORK AND SINZ (2013)).

Syntactic Inconsistency Syntactic consistency guarantees, that a model conforms to its language definition, usually specified by a meta-model.

Semantic Inconsistency Semantic consistency requires for aspects captured in different views to be semantically consistent to each other. The semantic information a modeler derives while looking at multiple views of a model must be coherent.

Horizontal Inconsistency / Intra-Model Consistency Horizontal consistency, also denoted as intra-model consistency, refers to views representing the same abstraction level to be semantically consistent.

Vertical Inconsistency / Inter-Model Consistency Vertical consistency, also denoted as inter-model consistency, refers to views representing different levels of abstraction, e.g., models built in different phases of software development like design models and analysis models to be semantically consistent.

Table 1 summarizes the four classes of consistency by combining syntactic/semantic and horizontal/vertical orthogonally in pairs. The concrete examples are taken from the Semantic Object Model (SOM) method (see section 7.2 for a thorough introduction to the SOM method).

[12] The effort is also documented by a specialized workshop on *consistency problems in UML-based software development* HUZAR ET AL. (2005) that took place from 2002 to 2005.

The importance of consistency resulted in several PhD projects specifically designed to investigate on possibilities on identifying and codifying semantic correspondences between multiple views. The interested reader may be referred to the dissertation thesis of DIJKMAN (2006) and HESSELLUND (2009). Also, the work of Romero et al., introducing a graphical modeling method for correspondences based on Object Constraint Language (OCL), Query View Transformation (QVT) (OBJECT MANAGEMENT GROUP (OMG), 2015) and Answer Set Programming (ASP) (ERAMO ET AL., 2008; ROMERO ET AL., 2009) might be interesting.

Table 1: Examples of consistency problem classes in the Semantic Object Model

	Syntactic	**Semantic**
Horizontal	A Business Object in the Object Decomposition Schema and the corresponding Business Object in the Interaction Schema should be consistent.	The Business Object responsible for the execution of a Task in the Task-Event Schema should be consistent to the corresponding Business Object in the Object Decomposition Schema
Vertical	A Business Transaction in the Transaction Decomposition Schema should be consistent to the corresponding transaction specific Objecttype in the Schema of Conceptual Classes	A internal event in the Task-Event Schema should be transformed into a corresponding interacts_with relationship in the Schema of Conceptual Classes.

2.3 Conceptual Modeling

Conceptual modeling approaches foster specificity of models by providing modeling concepts, derived from the application domain (MYLOPOULOS, 1982). Therefore, the abstraction level is raised to the level the modeler is familiar with. He or she creates the mapping of some real world phenomenon to the model by reusing concepts of the real world. The resulting models, referred to as conceptual models, act as knowledge basis, pursuing documentation and communication goals. *"A conceptual model of a domain is created by a (re-)constructing act of abstraction of domain concepts, i.e., concepts of a domain which are deemed relevant for a particular purpose. Conceptual models are designed by means of dedicated modeling languages that offer language concepts aimed at appropriately taking into account relevant perspectives on a domain"* (FRANK ET AL., 2014, p. 1).

A strength of conceptual modeling, by contrast to informal natural language or graphically sketched specifications, is its sound formal notation which enables to focus on the semantics of the real world phenomenon to be mapped. Compared to formal mathematical notations, conceptual modeling intends to be performed by humans and processed and interpreted by humans.

"Thus, conceptual models are considered to be linguistic (re-)constructions that are capable to purposefully structure and to clearly describe complex issues" (FRANK ET AL., 2014, p. 2) of some real world phenomenon. Consequently, conceptual models serve as a communication means in order to establish an inter-subjective understanding.

2.4 Model-driven Development

The goal of Model Driven Development (MDD) (and Model Driven Engineering (MDE)) approaches is to enable complex software system specification to take place at a higher level of abstraction. This goal is achieved by using models of different abstraction levels and systematically transforming them from abstract, conceptual levels into technical, platform-dependent models. To an extent, the final abstraction level is often a model of the software-specific implementation of the system by means of object-oriented models like UML class diagrams that are transformed into Java code.

Several sub-domains have been established in recent years, utilizing a model-driven development approach. Examples are Model Driven Architecture (MDA) (BOUCKÉ ET AL., 2010; OBJECT MANAGEMENT GROUP (OMG), 2003), Model Driven Software Engineering (MDSE) (VÖLTER ET AL., 2013). All of these approaches share the fundamental idea of deriving initial system implementations following a stepwise approach that continuously transforms conceptual models into implementation models. *"Model-driven engineering 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, p. 25).

The benefits of model-driven approaches, on a generic level, can be described by the following two aspects (SCHMIDT, 2006, p. 27):

Domain-specificity The developers can use domain-specific modeling languages in order to describe the concepts of the "key semantics of the domain" more appropriately and intuitively. Hence, the benefits of conceptual modeling, as introduced in section 2.3, can be facilitated.

Transformation capability The possibility to analyze and process the generated models automatically significantly eases the development process. In the final transformation stage, the models can be transformed into "XML deployment descriptions", simulation specifications or any other format required. In some approaches, models are also used to reflect on changes to the implementation; or the other way around, models are used to change the running system (cf. the workshop series *Models@runtime* at the yearly International Models conference).

However, besides all these positive aspects of MDD, some research challenges proposed by FRANCE AND RUMPE (2007) and criticism (HAILPERN AND TARR, 2006) are still open for investigation: *Modeling language challenges*; *Separation of concerns challenges*; and *Model manipulation and management challenges*. Interestingly, these research challenges can all be applied to multi-view modeling as they tackle the questions of how to specify viewpoints, how to determine the contents of and the relationships between viewpoints, and how to handle processing of views by the modeler, respectively.

2.5 Summary

The previous section introduced the methodical and conceptual foundations this thesis is built upon. Starting with an historical (STACHOWIAK, 1973) and a more formal (FERSTL AND SINZ, 2013) definition of the term *modeling*, both centering a mapping function between a selected area of the real world and a model to be build. Subsequently, the constituents of *modeling methods* and *meta modeling* have been introduced.

The major part of this section was dedicated to *multi-view modeling*. Starting with a brief history of multi-viewing in several domains, definitions of the important terms have been discussed and an emphasis has been put on the relationship types between viewpoints and the consistency issues raised by them.

The benefits of using *conceptual modeling* and *model-driven development* approaches then concluded this section. In the following, these foundations will be expected as given in order to proceed with a deeper investigation of the specifics of multi-view modeling methods and the conceptual design of multi-view modeling tools.

3 Related Work on the Conceptual Design of Multi-View Modeling Methods and Corresponding Tools

> *In current practice, enterprise architectures often comprise many heterogeneous models and other descriptions, with ill-defined or completely lacking relations, inconsistencies, and a general lack of coherence and vision. The main driver behind most of the needs identified above is the complexity of architectures, their relations, and their use. Many different architectures or architectural views co-exist within an organization. These architectures need to be understood by different stakeholders, each at their own level. The connections and dependencies that exist among these different views make life even more difficult. Management and control of these connected architectures is extremely complex* (LANKHORST, 2009)
>
> *(Marc Lankhorst)*

The following section is dedicated to present related work and discuss it in the context of this thesis. The selection of the presented approaches is based on a literature review, aiming at identifying approaches that support: i) the specification of multi-view modeling methods, and/or ii) the conceptual design and development of corresponding modeling tools.

3.1 Multi-View Systems Modeling

A multi-view modeling approach has been early adopted within the systems modeling domain. The Reference Model for Open Distributed Processing (RM-ODP) (UNION, 1997), published by the International Standards Organization (ISO) and the International Telecommunication Union (ITU) in 1997 introduced five viewpoints on a distributed architecture: *enterprise, information, computational, engineering, and technology* (RAYMOND, 1995). Each viewpoint *"is an abstraction that yields a specification of the whole system related to a particular set of concerns"* (UNION, 1997, p. 9). Although the standard has been used in practice, tool support was not given at that time due to the complexity of the multi-view architecture of the RM-ODP.

In 2004, Akehurst pointed to this lack of tool support by emphasizing on the positive aspects on usability and effectiveness such tools would have (AKEHURST, 2004)[13]. However, as

[13] Parts of that work are based on former publications investigating the development of tool support for the OMG's model-driven architecture (AKEHURST AND KENT, 2002; AKEHURST ET AL., 2003)

a formal specification of the viewpoints, the graphical visualization of the elements, and the correspondences between the viewpoints was lacking, developing a multi-view modeling tool for RM-ODP has been considered an extremely challenging task. In his paper, Akehurst started with the definition of necessary steps, constituting a procedural approach towards a conceptual design for a multi-view modeling tool for the RM-ODP. He raised notice to three parts such an approach should comprise AKEHURST (2004):

1. Precise definitions of the *viewpoint language concepts* - beyond the textual descriptions currently given - i.e., meta models.

2. Precise specification of the *correspondences between concepts* in each viewpoint, along with suggested mechanisms for specifying these - i.e., a Correspondence Specification Viewpoint.

3. Precise specification of *example notations* for each viewpoint. Perhaps both, graphical and textual.

Moreover, Akehurst already motivated what he called a *"full example system design"*, that should *"illustrate the intended use of the framework"* (AKEHURST, 2004). This can be considered a first attempt to define a procedure model for the conceptual design of multi-view modeling tools. The approach was specifically developed for the requirements given by the RM-ODP framework, however, most of the requirements and suggestions are on an abstract level. Tool designers and RM-ODP experts have been suggested as the ones who constitute a set of requirements and transform them into a technological model, i.e., used for model-driven development of corresponding tools. In AKEHURST (2004), the author also considers the behavioral aspect of interacting with multi-view models by introducing correspondences between concepts of different views. Thus, allowing to check the consistency between the views and derive mechanisms that could be implemented to automatically preserve a consistent model state.

3.1.1 Multi-View Modeling with SysML

The following approach originates from the domain of embedded systems engineering. SHAH ET AL. (2010) introduced a methodical approach for the development of multi-view modeling methods based on OMG's Systems Modeling Language (SysML) (OBJECT MANAGEMENT GROUP (OMG), 2012b). All viewpoints are generated from a common base SysML model. Viewpoint consistency is enabled using graph transformations. Shah et al. propose a methodical approach comprised of three steps (SHAH ET AL., 2010, p. 585ff.):

1. *Formal definition of the domains involved in the system through meta models*
   In the first step of the approach, the authors suggest to define a formal specification of the constituents of a viewpoint and the rules for combining them. This formal specification is realized by meta models created with the MOFLON[14] modeling tool (AMELUNXEN ET AL., 2008; LEBLEBICI ET AL., 2014). Moflon generates meta models that comply to the MetaObject Facility (MOF) (OBJECT MANAGEMENT GROUP (OMG), 2011b) standard.

2. *Customization of SysML through profiles to enable domain-specific modeling*
   The second step of the approach transforms the meta models into a common basis. This basis is the SysML modeling language. Consequently, each domain-specific meta model is transformed into a SysML profile. Specifics of each domain are captured using stereotypes.

3. *Model transformations to generate domain-specific views from SysML*
   The last step considers the generation of the multiple views from the common model. The authors suggest using *story diagrams* (FISCHER ET AL., 2000) to visually specify the transformation rules. *"These rules are Triple-Graph Grammar (TGG) like in that they use a correspondence graph that captures the relationships between source and target model"* (SHAH ET AL., 2010, p. 593).

3.1.2 Discussion

Akehurst concludes by discussing some of the major problems that need to be tackled in the future, e.g., the specification language for the viewpoint language, the correspondences, the notation, and the technology model. The author identified important research questions but the answer is left for further research. The recommendations stay on a superficial and informal level. The created ideas are prototypical realized using languages of the Object Management Group (OMG) that obviously not cover the semantics of the multi-view domain adequately. A modeling environment, supporting the approach is wished for by the authors - however such a tool has never been realized.

The approach of Shah et al. concentrates on the syntactic integration of multiple views and is strongly biased by (and limited to) the systems modeling domain. The approach lacks at considering the needs of the human beings, creating, studying, and processing the multi-view models as well as the procedure of interacting with multiple views. Moreover, the viewpoints need to be realized as an extension of the OMG SysML modeling language meta model. Due to the application domain, the approach is designed for tool developers with highly technical skills in programming languages and knowledge about how to utilize Triple Graph Grammars.

[14] `http://www.moflon.org`, last checked: 2015-03-01

3.2 The Marama Meta-Toolset

Several conceptual foundations of the Marama meta-toolset are grounded on former research of the authors on multi-view editing environments (GRUNDY AND HOSKING, 1993) and a tool that enables multi-view programming of object-oriented software systems (GRUNDY ET AL., 1991). A Marama model project *"contains one model instance with multiple view (diagram) instances, all kept consistent with one another"* (GRUNDY ET AL., 2013, p. 493f.).

In the current form, the Marama environment supports the development of multiple Domain-Specific Visual Language (DSVL) editors by providing a set of Eclipse[15] plug-ins: The **Meta-model Designer** utilizes an Extended Entity Relationship (EER) notation for the simplified specification of the meta model. Central modeling concepts in the EER notation are *entities*, *attributes*, *associations*, and *event handlers*. A **Shape Designer** enables the model-driven specification of graphical visualizations for the meta model elements. The definition of viewpoints (the authors call them view types), consisting of a subset of all visual elements of the meta model, is performed using the **View Type Designer**. This designer maps elements of the meta model to the graphical visualization, thereby specifying a viewpoint. Moreover, attribute mappings between viewpoint elements and concepts of the meta model are defined using a **Viewpoint Wizard**. This enables automatic generation of consistency-preserving mechanisms for these mappings by means of OCL constraints.

The behavioral aspects of a DSVL tool can be specified with a visual editor for the OCL constraints. Moreover, the **Event Handler Designer** enables the imperative specification of the behavior by defining *event-condition-action* and *event-query-filter-action* rules. The substantial set of behavioral specifications, including consistency constraints, viewpoint dependencies, diagram layout, and visual event handlers are condensed in the MaramaTatau extension of the Marama EER meta model designer (LIU ET AL., 2007). Another Marama extension, Kaitiaki, which is based on the visual event processing language Serendipity (GRUNDY AND HOSKING, 1998), enables the graphical specification of complex behavioral rules that are triggered by events, e.g., change of position, change of attribute values.

One strength of the Marama meta-toolset is its *"liveness characteristic"* (GRUNDY ET AL., 2008), i.e., changes on the specifications are immediately reflected by any reopened model instance. The goal of the approach is to enable the specification of DSVL tools on a more appropriate abstraction level, thereby enabling also non-programmers the development of such tools in an efficient way. According to GRUNDY ET AL. (2013), the approach is directed to two user groups: First, domain modelers who actually want to model within a concrete domain using a domain-specific visual language. Second, tool developers who want to create DSVL modeling tools following the model-driven development approach. Further extensions and plug-

[15] `https://eclipse.org/`, last checked: 2015-04-03

ins to Marama have been implemented, e.g., *MaramaSketch* (GRUNDY AND HOSKING, 2007), supporting sketch-based modeling, and *MaramaThin* (CAO ET AL., 2005; ZHAO ET AL., 2006), enabling collaborative modeling using a web browser.

3.2.1 Viewpoint Update Records

(GRUNDY AND HOSKING, 1993, p. 34) introduced the usage of *update records* for each viewpoint. These update records store all modifications on one view. Consistency between the multiple views can then be ensured in two ways: (1) *data-driven*, i.e., immediate update of all affected views; or (2) *demand-driven*, i.e., update of affected views when the modeler decides it.

Consistency between the views is therefore ensured by sharing the update records between all views. Each view then locally detects, whether changes listed in the update record require changes on it. If changes are required, the viewpoints transform the changes described in the update record into corresponding changes on their own. The transformation is important, as different viewpoints may require different changes in order to be kept consistent. However, the authors do not describe in detail how the tool recognizes changes, which types of changes are recognized, and how different changes on the multiple viewpoints can be specified. Due to the viewpoint-specific handling and interpretation of updates, the approach also lacks scalability as the effort for specification increases witch the number of viewpoints.

3.2.2 Discussion

The Marama tool family provides a rich set of tailored tools, each specialized for a specific task in the model-driven development of multi-view modeling tools. However, a quite skilled programmer is required to use all the different tools and specification languages. The tool is targeting at the development stages, i.e., the implementation of a multi-view modeling tool for an already specified modeling method. Although the Marama tools provide a more sophisticated, i.e., higher abstraction level, a method engineer might easily loose the overview due to the complex interplay of the numerous tools and specification languages utilized by the approach. The Marama tools can be used to conceptualize a multi-view modeling tool - however, the tools are specialized for the development of Eclipse modeling editors.

3.3 Orthographic Software Modeling

The Orthographic Software Modeling (OSM) approach *"is based on the idea of creating a Single Underlying Model (SUM) that contains all information about the system currently available, and separate view models that contain the information to be displayed in specific views of*

the system“ (ATKINSON ET AL., 2010, p. 210). The OSM approach replaces the need for defining inter-view consistency constraints for each pair of views with the requirement of specifying two ATLAS Transformation Language (ATL) [16] transformations.

The first transformation defines how viewpoints can be derived dynamically from the SUM, whereas the second transformation specifies, how changes on a viewpoint a transformed into changes applied to the SUM. Whereas the former needs to be specified for each considered viewpoint, the latter is only needed, if the viewpoint is also editable.

(ATKINSON ET AL., 2010, p. 211) define a view as *“a normal model which just happens to have been generated dynamically for the purpose of allowing a user to see the system from a specific viewpoint“*. The definition of the viewpoints is supported by a **view designer**, enabling the selection of concepts, relations, and attributes of the single underlying model to be considered by a viewpoint. Moreover, every viewpoint is assigned one of the multiple, hierarchically structured dimensions. The structuring of a viewpoint along these dimensions enables the modeler to navigate along the hierarchy, thereby interactively switching to other viewpoints of the system. The set of hierarchies is not fixed, i.e., it can be customized to map specific requirements. Initially, the dimensions *composition*, *abstraction*, and *projection* have been defined (ATKINSON AND STOLL, 2008, p. 95). The benefit of utilizing the orthographic projection metaphor for the viewpoint specification is twofold: First, the metaphor is intuitively understandable by modeler and model users. Second, applying the orthographic projection operator results in viewpoints with minimal semantic and syntactic overlapping areas.

3.3.1 Flexible Viewpoints

(BURGER, 2013; KRAMER ET AL., 2013; GOLDSCHMIDT ET AL., 2012), introduced flexible, not predefined viewpoints for model-driven development of software systems. An emphasis is also on the correspondences between the viewpoints and mechanisms for ensuring consistency. An initial set of correspondence rules is generated automatically during the specification of the viewpoints. These rules can be further processed (i.e., refined) by the user.

Like for OSM, the focus is on using the multiple viewpoints for software development. The specification of the viewpoints and the modeling procedure, both from a method owner perspective, are not in the focus. Moreover, implementation experience is required in order to specify complex relationships and consistency rules between multiple views.

[16] ATLAS Transformation Language homepage: `http://www.eclipse.org/atl/`, last checked: 2014-05-27

3.3.2 Discussion

Recently, the OSM approach has been finalized and published together with a modeling tool environment supporting its application (ATKINSON ET AL., 2013a,b). The focus of the approach is on software development. The early development stages of multi-view modeling methods are not targeted. However, due to the long lasting development of the OSM approach and the supporting tool environment - first articles have been published in 2003 (ATKINSON AND KÜHNE, 2003) - both, approach and tool, are in a mature status and have proven to be helpful in the model-driven development of software systems. One drawback of the original OSM approach was the fact that all viewpoints needed to be predefined in order to create the invariant single-underlying model (SUM). The flexible viewpoints approach omitted this drawback, however still tackling the software development domain with a focus on the late development stages.

3.4 Hybrid Modeling

In 2013, Karagiannis and Schwab introduced the *hybrid modeling approach*. The approach is motivated by the need for "modular construction systems" and "open modeling approaches", covering the "divers business requirements in a fast moving environment" thereby amplifying flexibility (KARAGIANNIS AND SCHWAB, 2013, p. 3). The authors define two major requirements for their hybrid modeling approach: (1) *"the result of hybrid modeling is a modeling method that falls in the category of a graphical semi-formal modeling method based on a meta-modeling approach"*, and (2) *"the hybrid modeling method is implemented and offered in form of a modeling tool"*. Due to this focus, the authors also define a *"deliberate procedure comprising different steps"* (KARAGIANNIS AND SCHWAB, 2013, p. 1) towards the implementation of a modeling tool for a hybrid modeling method - in the following referred to as *conceptualization*.

3.4.1 Conceptualization Life Cycle

The process of implementing a hybrid modeling tool in accordance to a hybrid modeling method is divided into the phases **create**, **design** and **compile**. The conceptualization is dependent on the ADOxx meta modeling platform (see section 7.3.1 for a detailed introduction to the ADOxx platform). Figure 19 illustrates the conceptualization by means of a life cycle.

In order to foster the implementation of hybrid modeling tools, the authors state, that *"the better the resulting hybrid modeling method is conceptually composed, the smoother the transformation of the concepts to the codes can be done"* (KARAGIANNIS AND SCHWAB, 2013, p. 3). The integration of different building blocks of multiple modeling methods requires the analysis of the different abstraction levels. Three *"states in a way meta models vary"* (KARAGIANNIS AND SCHWAB, 2013, p. 3f.) have been identified: *vertically*, if they show different

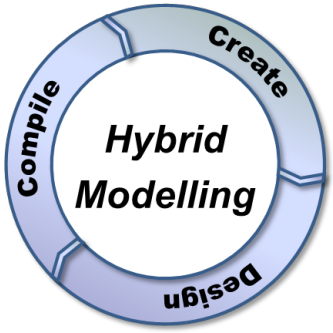


Figure 19: Hybrid modeling conceptualization life cycle (KARAGIANNIS AND SCHWAB, 2013, p. 5)

levels of abstraction; *horizontally*, if the concepts of the building blocks are on the same abstraction level but they describe semantically different aspects; and *both*, if a combination of the former two states is given. These different states are explicitly considered in the conceptualization of a modeling tool, described in the following:

3.4.1.1 Create Phase

The first phase of the hybrid modeling approach *"is related to the application scenario and the need of the user and refers basically to the selection process the user performs for identifying the existing modeling concepts within the hybrid method"* (KARAGIANNIS AND SCHWAB, 2013, p. 4f.). Basically, the first phase collects all scenarios that should be supported by the modeling method to be built. These scenarios lead to the building blocks that need to be integrated in the hybrid method. Having the building blocks defined, the relationships and dependencies between the concepts of different modeling methods need to be specified. *"Depending on the maturity of the underlying building blocks the description can be of formal but also informal kind or can for example include the definition of a consistent meta model of the hybrid modeling method on a sole conceptual level"* (KARAGIANNIS AND SCHWAB, 2013, p. 5).

The creation of a hybrid modeling method is strongly aligned to the components of modeling methods introduced by Karagiannis (cf. Figure 8). The hybrid modeling approach therefore analyses all three major parts of a modeling method: *modeling language*, *modeling procedure*, and *mechanisms & algorithms*.

3.4.1.2 Design Phase

In the second phase, the input of the creation phase in combination with the intended platform for the development of the hybrid modeling tool is used to elaborate *"the preconditions for the later implementation respectively customization phase"* (KARAGIANNIS AND SCHWAB, 2013,

p. 6). The development platform, and the functionality it provides, influences the design decisions of the method engineer in the design phase and the implementation phase. The purpose of the design phase is the definition of two meta models, a *conceptualization meta model* and an *implementation meta model*. The separation of those two meta models reflects the platform-independent integration resulting from the create phase and the platform-dependent realization in the design phase. This realization is performed by mapping the concepts of the hybrid modeling method meta model to the generic concepts provided by the ADOxx platform by means of the ADOxx meta meta model.

Due to the fact, that the hybrid modeling approach is directed towards the implementation of graphical modeling tools, the *notation* of the concepts included in the hybrid method plays a mature role. Karagiannis and Schwab identified two combinations that lead to inconsistencies and therefore to design decisions for the method engineer (KARAGIANNIS AND SCHWAB, 2013, p. 9): (1) multiple, inconsistent notations due to the existence of more than one concept that need to be merged; and (2) missing notations, as the precise and unambiguous specification of the notation might be missing for one or more modeling concepts.

Along with the discussion about the assimilation of the possibly differing notations goes the consideration about the *semantic alignment* of different concepts. In order to decide about the semantic mapping between the building blocks, the authors propose the usage of a semantic comparison table. On each axis of this table, the concepts of one modeling method are positioned. For each cell of the resulting table, i.e., for each pair of concepts of modeling methods, the semantic correspondence is defined. The scale ranges from (−), *"not applicable - comparison of modeling class and relation class"*, over (!=), *"unlike, does not correspond at all"*, and (∼), *"natural language description and use show similarities"*, to (1:1), *"identical in their natural language description and use"* (KARAGIANNIS AND SCHWAB, 2013, p. 10). Table 2 shows an excerpt of the comparison of i* and BPMS modeling method concepts.

The third mapping task of the design phase (after mapping the notations and the semantics) is concerned with the *syntactic mapping* of modeling concepts derived from different modeling methods. The syntax *"describes the dependencies and constraints in between the modeling concepts and is furthermore represented in the description of the properties of these in form of attributes"* (KARAGIANNIS AND SCHWAB, 2013, p. 10). The syntactic mapping therefore must define the integration of the attributes of semantically identical concepts (e.g., identified by a 1:1 value in the comparison table between these concepts).

3.4.1.3 Compile Phase

In the compile phase of the hybrid modeling approach, the concrete realization of the designed hybrid modeling method is performed using the ADOxx meta modeling platform. As several steps do not require an implementation but a customization of predefined platform functionality,

Table 2: Semantic comparison of i* and BPMS modeling method concepts (KARAGIANNIS AND SCHWAB, 2013, p. 10)

Modeling Concepts		i* classes and relations				
		Actor	Agent	Position	Association Link	Dependency Link
BPMS classes and relations	Organizational unit	∼	!=	!=	–	–
	Performer	∼	∼ almost 1:1	!=	–	–
	Role	∼	!=	∼	–	–
	Position	!=	!=	!=	–	–
	Is subordinated	–	–	–	∼	!=
	Belongs to	–	–	–	∼	!=

Legend: !=, unlike, does not correspond at all; **1:1**, identical in their natural language description and use; ∼, natural language description and use show similarities; –, not applicable - comparison of modeling class to relation class

this phase is also referred to as customization phase. The main task is the decision about the most appropriate representation of the hybrid modeling method to the modeler, e.g., using a modeling app for mobile devices, a locally installed stand-alone application, or a web-based client-server modeling environment.

The compile phase does significantly depend on the development platform, e.g., a meta modeling platform. The concepts and functionality provided by the platform must be used in order to implement all components of the designed hybrid modeling method. The result of the compile phase is a complete hybrid modeling tool, enabling human beings to create, interpret, and process the generated models in an appropriate way.

3.4.2 Discussion

The approach concentrates on the combination of *"building blocks"* of modeling methods in order to generate flexible hybrid modeling methods. Especially the create phase show a significant amount of similarity compared to the *Modeling Scenario* provided in the MUVIEMOT method (cf. section 6.3.1). Both consider the specification and integration of modeling views that optionally originate from different modeling languages or methods. The emphasis of the hybrid modeling approach is on the integration of given modeling methods by stepwise integrating the building blocks of the modeling method framework illustrated in Figure 8.

The modeling procedure of the resulting hybrid modeling method is not considered appropriately by the approach. This is a major drawback, as the way of carrying out multi-view modeling plays a mature role when it comes to utility and efficiency of the tools. It seems unlikely, that merging different modeling methods does not require the integration of the corresponding modeling procedures and mechanisms & algorithms. Lastly, the hybrid modeling approach targets the implementation of modeling tools given a predefined set of modeling methods. It is not clear, how the multiple modeling methods are transformed into multiple viewpoints of the tool conceptually.

3.5 Hybrid Multi-View Modeling Approach

CICCHETTI ET AL. (2011, 2012) proposed a **hybrid multi-view modeling approach** based on the Eclipse Modeling Framework. The approach is strongly aligned to the technologies of the Eclipse platform. It uses *model to model* and *model to text* transformations to manage consistency and to generate Eclipse EMF editors, respectively. The approach is built on top of a comprehensive, single meta model. Therefore, the authors argue, the generated views, derived by a projection on this meta model, are *"consistent by construction"* (CICCHETTI ET AL., 2011, p. 1). However, this consistency only covers one-to-one dependencies between elements, i.e., the same Ecore concept being considered in multiple views. More complex dependencies, e.g., between different classes and different attributes of the same or different classes, are not considered.

3.5.1 Change propagation

The change propagation is performed via the centralized model. First, changes are applied to this Ecore meta model, then all affected views are updated accordingly. The update mechanisms use *model to model transformations* based on difference meta models between views and the central meta model. By comparing different models, difference models are created as instances of the difference meta models. A more detailed description on the difference meta models and the synchronization mechanisms can be found in (CICCHETTI ET AL., 2007) and (CICCHETTI ET AL., 2012), respectively.

3.5.2 Discussion

The domain of the approach is the development of EMF modeling editors using multiple, interrelated views that are derived from a common Ecore meta model. An emphasis is on the consistency by construction and the specification of editing rights for each concept in each view. A major strength of the approach is the **view creation editor** that allows flexible specification of

views by selecting the classes and attributes of the underlying meta model. Additionally, the automatically generated model comparison transformations and model to model synchronization transformations are powerful for automatic handling of rudimentary consistency constraints. Due to the single underlying meta model, the view creation editor, and the consistency mechanism, the hybrid multi-view modeling approach is quite similar to the Orthographic Software Modeling approach introduced by Atkinson (see section 3.3).

Major drawbacks of the approach are the limitations given by the comprehensive underlying meta model and the rudimentary specification of consistency constraints on the meta model level. Moreover the approach does not consider the modeling procedure. Graphical visualization is only provided by the standard tree-based notation of EMF. Due to the strong relation to the Eclipse platform and its textual specification languages, the approach is aiming at tool developers with implementation experience.

3.6 Ontological Multi-View Modeling

Like meta modeling, the creation of ontologies provides some structuring of the domain it is created for. By contrast, ontologies don't focus on the syntactic aspect of a modeling language. An ontology can be used to structure the semantic area of a domain. Using the Web Ontology Language (OWL)[17], one is able to define classes, properties of classes, and relationships between classes using semantic constructs.

In the modeling domain, ontologies can be also used as an integration means. Providing a group of modelers not only a modeling tool with the syntax of the modeling language but also with one ontology, constraining all semantic terms in the application domain, enables the comparison and inter-subjective understanding of the created models. Each modeler is forced to refer e.g., the name of a modeled activity in a process model to a concept of the ontology. Ambiguities and misinterpretations are limited by the usage of the ontology. Also, collaboration of the modelers is endorsed by the ontology as the models are built *"using the same language"*. Due to all these positive aspects, ontologies haven been used in very different domains such as process modeling (THOMAS AND FELLMANN, 2007), enterprise modeling (FOX, 1992), (model-driven) development of information systems (HAPPEL AND SEEDORF, 2006; KNUBLAUCH, 2004), and integrating different domain-specific modeling languages (WALTER AND EBERT, 2009; WALTER ET AL., 2009). In 2012, Fill introduced the SeMFIS method and a corresponding modeling tool[18] that uses ontologies in order to bridge the semantic gap between different conceptual modeling languages (FILL, 2012). The tool moreover allows the semantic querying of created models following a meta modeling approach.

[17] W3C. OWL 2 Web Ontology Language, `http://www.w3.org/TR/owl-features/`, last checked: 2014-05-02

[18] The SeMFIS modeling tool: `http://www.omilab.org/web/semfis`, last checked: 2015-04-03

Combining ontologies with meta modeling is also a promising research field. In the context of multi-view modeling tools, KUSEL ET AL. (2012) describe an approach that combines several modeling views using a higher abstraction level, i.e., combining them by referring their concepts to concepts defined by one comprehensive ontology model. Using the OWL language, one is able to define a comprehensive ontology and then define the views. Out of these views, Eclipse meta models in the Eclipse Modeling Framework (EMF) are generated. The transformation is based on a meta modeling approach by mapping the meta concepts of the OWL ontology language (i.e., the Ontology Definition Metamodel (ODM)) to the meta concepts of Ecore (i.e., the Ecore meta meta model). This allows the automatic transformation of all instantiated OWL ontology models into Ecore meta models.

3.6.1 View integration

Integration of, and synchronization between the views is provided by the same domain meta model (specified in OWL). Moreover, OCL constraints can be implemented in order to introduce more complex constraints between the views. The tool also supports the specification of multiple graphical representations for one concept. The approach proposed by Kusel et al. does in some way define a generic procedure, i.e., first create the ontology, then transform the views and automatically generate the Ecore meta models complemented with the constraints that can be automatically derived. The modeling tool then provides synchronization mechanisms between the more technical Ecore meta model and the OWL ontology. Figure 20 illustrates the process of generating Eclipse modeling tools with the approach.

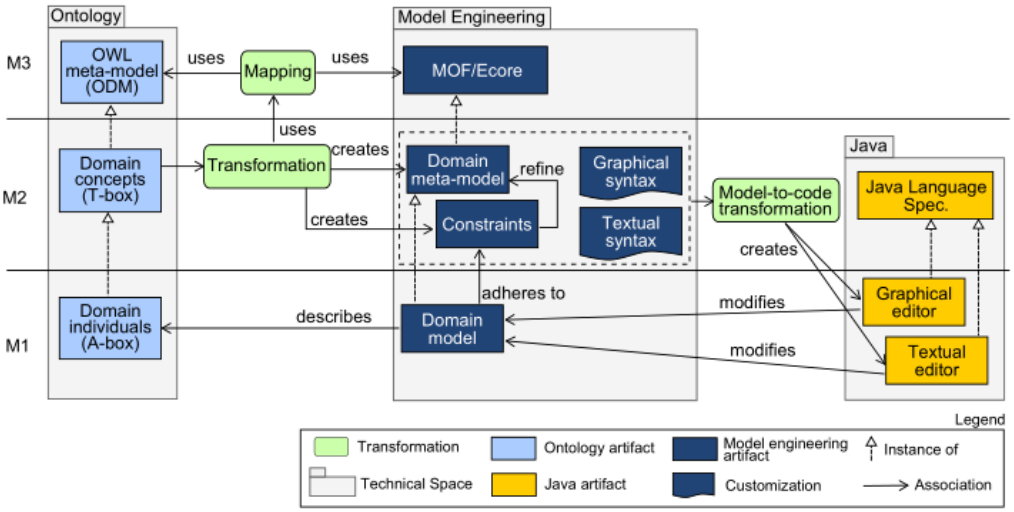


Figure 20: Generation of modeling tools based on meta models and ontology models (KUSEL ET AL., 2012, p. 47)

3.6.2 Discussion

The approach does not consider the early development stages properly. The multi-view modeling procedure is not covered. The approach seams promising for a given scenario with already defined domains and meta models. Moreover, a technically skilled person should apply it due to its manifold technical specification languages used, e.g., Ecore, Java, OCL, Xtext. Generally, the approach allows the generation of multi-view modeling tools based on the Eclipse Modeling Framework. It serves two intended users, one who creates the domain-specific ontology model and a very skilled developer who transforms the ontology into Ecore meta models and complements them with further constraints, graphical representations, and tool functionality.

Especially, the specification of the modeling procedure in the context of a multi-view modeling tool is not considered appropriately. The approach relies these procedural aspects to Java and OCL, which, in both cases, is neither an appropriate abstraction level nor specifically aligned to the characteristics of a multi-view modeling tool. The approach can be used to couple and semantically integrate different meta models using a common ontological model. This coupling enables analysis, interpretation and transformation of views created with different modeling languages.

3.7 IEEE 42010 Architecture Modeling

The core architecture description of the IEEE standard 42010 is illustrated in Figure 15. In order to foster the application and ease the use of the standard, the IEEE (in the person of Rich Hilliard) published a template for the specification of architecture viewpoints according to the standard 42010. This template *"defines a set of "slots" or information items to be elaborated by the architect using the template to define and specify a viewpoint"* (HILLIARD, 2012, p. 1). For each slot, the template provides a brief description of its semantics and content, and some guidance for the generation of that content. The slots are divided into the categories optional and mandatory. The components of the latter category are now briefly described (HILLIARD, 2012, p. 3ff.):

Viewpoint Name Definition of a name for the viewpoint. Additionally, *"synonyms or other common names"* (HILLIARD, 2012, p. 3) can be specified.

Overview This slot provides an informal description of the architecture viewpoint by means of natural language. The key features of the viewpoint can be defined in order to enable efficient understanding of the intended focus of the viewpoint.

Framed concerns and typical stakeholders The third mandatory slot uses the overview slot and introduces a precise description of the viewpoints characteristics. This slot includes the concerns that should be framed by the viewpoint (e.g., specified in the form of

questions to the system of interest, the viewpoint should be able to answer) together with the intended stakeholders, the viewpoint is directed to (e.g., users, operators, managers). Optionally, *anti-concerns* can be defined in order to provide a clear separation between different architectural viewpoints and foster the search process for a specific viewpoint.

Model kinds As the conceptual model indicates, multiple model kinds can be part of an architecture viewpoint (cf. Figure 15). The model kind slot defines for each of the viewpoints' model kinds its *"notations, conventions, and rules for that kind"* (HILLIARD, 2012, p. 5). Accordingly, a precise specification of each model kind follows by defining the following criteria for each model kind individually: *Name*, for the identification of each model kind; *Conventions*, covering languages, modeling techniques, notations, tools, methods and operators, used by the architect to create a model kind (e.g., including meta models, constraints, ontologies, or templates); *Operations*, optionally operations on a model created accordingly to the model kind can be specified (cf. the "Operations on Views" slot for more details); *Correspondence Rules*, rules can be defined that enable e.g., consistency checking between different model kinds (cf. the "Correspondence Rules" slot for more details).

Operations on Views This slot specifies methods that can be applied to a model kind. Four operation categories are distinguished:

- **Construction Methods**: All methods guiding the construction of a view by means of a viewpoint. Different kinds of methods are subsumed under this category, i.e., sequence of actions that need to be performed in order to create valid models and heuristic construction techniques.
- **Interpretation Methods**: Operations guiding the reader by delimiting the information defined in the view.
- **Analysis Methods**: The operations of this category aim at analyzing the architecture model from a certain architecture viewpoint and the processing of these results.
- **Implementation Methods**: All operations guiding the processing of an architecture view "to design and build systems" (HILLIARD, 2012, p. 7).

Correspondence Rules *"Correspondence and correspondence rules are used to express and enforce architecture relations such as composition, refinement, consistency, traceability, dependency, constraint and obligation"* (IEEE, 2011, p. 7).

Sources This slot allows the specification of any sources, relevant for the architecture viewpoint, e.g., a version history, authorship, bibliographic information, scientific publications, references.

Discussion

The international standard ISO 42010:2011 in the current version provides a rich set of slots enabling a comprehensive specification of an architecture description. The standard integrates different architecture viewpoints by explicitly considering the correspondence between these different viewpoints. Moreover, the standard decomposes an architecture description into several architecture viewpoints, which itself are composed of different model kinds, each defined e.g., by a different modeling method and focusing on a certain aspect of the architecture viewpoint. The standard also includes slots for the definition of operations on model kinds therefore enabling the specification of the modeling procedure. Additional information documenting the interpretation, usage, and processing of the viewpoint concludes the viewpoint specification.

Although the standard provides structuring functionality and guidance during the specification of architecture descriptions, the specification itself is on an informal level. The consistent inter-subjective understanding of the standard considerably depends on the person creating the specification. Most viewpoint slots are specified imprecisely. The standard lacks at covering the specifics of modeling methods in the definition of the views, the overlaps, the consistency rules, and the modeling operations. However, due to the openness an imprecision of the standard, it is possible for an experienced requirements engineer to introduce most of these specifics in different slots. The transformation of the requirements into software systems allowing the creation of architecture descriptions is not targeted by the standard.

3.8 Summary

Although the preceding section provided a number of approaches related to this thesis, the selection does not claim to be complete. Manifold slightly differing approaches can be found in the literature trying to break down the complexity of composing multiple views and/or designing and developing multi-view modeling tools. The approaches included in this section aim to establish a comprehensive overview of relevant approaches and proposed solutions.

Consequently, this section presented the most relevant approaches proposed to enable the specification of multi-view modeling methods and the conceptual design and development of appropriate modeling tools. The approaches share three major deficits:

1. They lack in formalization. Most approaches stay on a conceptual and mereological level.
2. Most approaches concentrate on the development of systems by integrating given viewpoints or modeling methods. They do not consider the early conceptualization steps.
3. Most of the approaches do not consider the specifics of multi-view modeling methods in general and the modeling procedure in particular. By contrast, they focus only on some aspects like syntactic integration, semantic integration or viewpoint relationships.

The identified shortcomings establish the motivation for the main contributions of this thesis, presented in the following. Consequently, section 4 concentrates on the benefits and possibilities of a formalized modeling method specification. Section 5 then provides a thorough investigation on the specifics of conceptual modeling using integrated multiple views. In section 6, the findings are comprised into the requirements specification, conception, and development of a conceptual modeling method, aiming at the model-driven conceptual design of multi-view modeling tools.

4 Formalized Specification of Modeling Methods

> *Another important property of an enterprise modelling language - and of any modeling language - is a formal foundation, which ensures that models can be interpreted in an unambiguous way and that they are amenable to automated analysis.* (LANKHORST, 2009, p. 87)
>
> *(Marc Lankhorst)*

When designing and operating advanced knowledge and work systems that integrate emerging technologies with existing business processes for leveraging additional value for enterprises, one is confronted with high complexity due to the numerous dimensions that need to be taken into account (ALTER, 2008). Besides environmental factors like globalization of businesses, fierce international competition, and increasing employee mobility, these dimensions also include technical aspects such as rapid and frequent changes in information and communication technologies (cf. MAIER (2004)) and the increasing integration of different computer systems. This integration forces not only a machine-processable knowledge base, but also an inter-subjectively understanding.

In order to manage this complexity and support the communication between users and developers, it is common to revert to *conceptual enterprise modeling methods*. These methods permit to represent static and dynamic phenomena of systems prior to their implementation (WAND AND WEBER, 2002). Conceptual modeling produces a *"common understanding"*, therefore acting *"as a communication, analysis, and documentation tool for domain and IS requirements"* (MAES AND POELS, 2007, p. 702). Furthermore, it provides input for the system design process (MAES AND POELS, 2007; WAND AND WEBER, 2002). In addition, the models provide value themselves by acting as *knowledge bases* for answering queries, simulating behavior, performing reasoning, verification & validation, or generating executable code (FILL AND KARAGIANNIS, 2013).

For realizing this additional *model value* it is important to provide sound and inter-subjectively exchangeable foundations for the underlying modeling methods. This is required not only for ensuring the exact understanding of the structure and behavior of the modeling methods. It is essential for realizing the processing by machines in the form of algorithms and the inter-operability between different systems. As Meseguer and Preece state, *"the absence of formal specifications limits the capacity of knowledge-based systems"*, concluding that formal specifications can play a fundamental role in accomplishing *"adequate answers to issues such*

as correctness, completeness, robustness, precision, safety, and so forth" (MESEGUER AND PREECE, 1996, p. 321). Moreover, the usage of natural language specifications is not adequate as Meyer argues *"The main advantage of natural language texts is their understandability. One should concentrate on this asset rather than trying to use natural language for precision and rigor, qualities for which it is hopelessly inadequate"* (MEYER, 1985, p. 22). Thus, it becomes necessary to provide *formalized*, i.e., unambiguous specifications of modeling methods.

Some of the commonly used methods provide formal specifications right from the start, whereas for others, such formal specifications have been added later or are still not available - see e.g., the recent discussion centering around the current formal specification of UML (OBJECT MANAGEMENT GROUP (OMG), 2011d), structural and behavioral semantics (SEIDEWITZ, 2013), and formal notation specifications (OBJECT MANAGEMENT GROUP (OMG), 2012a). These current discussions show, that there exists a strong demand on formalization considering the UML, one of the most influential and utilized approaches for object-oriented specification of software-intense systems (ENGELS ET AL., 2001).

The following sections first investigate how components of modeling methods, referring to section 2.1.2 and summarized in Figure 8, can be specified in a formalized way. Based on the findings, a comprehensive analysis framework is derived which will then be applied to a selection of enterprise modeling methods, thereby discussing the degree of formalization in the respective specification of the methods. The analysis framework as well as the subsequent application of the framework concentrate on the process-related aspects of the methods. Several parts of this section have been published in (BORK AND FILL, 2014) and presented at an international conference. This section provides an extended version in two ways: 1) more analyzed methods, and 2) more comprehensive introduction/discussion of the methods.

4.1 Formalization of Modeling Methods' Specifications

This section covers the foundations for generally describing modeling methods. Therefore, the modeling language, modeling procedure, and mechanisms & algorithms are considered (cf. section 2.1.2). Subsequently, the spectrum of potentially formalized specifications of those aspects will be discussed. The goal is to describe how specifications of certain components of modeling methods can be defined in a formalized way. This results in a comprehensive analysis framework which can then be applied in section 4.2 in order to analyze enterprise modeling methods. The results of this analysis are then discussed in section 4.3.

The complexity of today's enterprise systems fosters the need for approaches that can handle the complexity and break it down into manageable parts for a human being. Over the last years, several enterprise modeling methods have been introduced in theory and practice trying to bridge that gap. Enterprise modeling methods divide the complexity of an enterprise by

providing dedicated views on that enterprise - e.g., views on the structure, behavior, processes, functions, IT systems, machines, personnel and organization of an enterprise.

A central aspect of almost any enterprise modeling method is the definition of the behavioral aspects of an enterprise by means of processes (SHEN ET AL., 2004). These processes are usually described using process-oriented models. They play a vital role for the enterprises which is why the design of work systems (cf. ALTER (2008)), supporting and realizing those processes, should be strongly aligned to the process models.

4.1.1 Related Work on the Analysis of Enterprise Modeling Methods

Several authors have analyzed selected enterprise modeling methods based on a given application domain or an intended usage scenario. LAKHOUA AND RAHMOUNI (2011) analyzed several enterprise modeling methods according to the domains they can be applied in, consistency, polyvalence, and simulation. SZEGHEO AND ANDERSEN (1999) investigated enterprise modeling methods based on their underlying approach, i.e., active knowledge modeling, process modeling, object-oriented modeling, and agent-based systems. They conclude, that every approach has its dedicated application area and by giving *"suggestions [..] how these approaches can be used for modeling the extended enterprise"* (SZEGHEO AND ANDERSEN, 1999, p. 8). Recently, BOCK ET AL. (2014) published an analysis of enterprise modeling methods based on their constituents. The authors propose a framework, comprising the following criteria: *Way of thinking:* methods were analyzed by investigating their background, goals, and central assumptions; *Way of modeling:* analyzing the semantics, syntax and notation the methods provide; and *Way of working:* investigating whether the methods utilize a modeling procedure and whether or not processes are specifically considered.

These references exemplify the current research approach around analyzing enterprise modeling methods, i.e., by investigating existing enterprise modeling methods according to their suitability for a given application domain or context. This section pursues a different, domain-independent approach by discussing and analyzing the degree of formalization in the specification of enterprise modeling methods' process-related aspects.

4.1.2 Proposition of an Analysis Framework

In order to classify modeling methods according to their degree of formalization an analysis framework is established in the following, comprising a set of criteria based on the generic components of modeling methods summarized in Figure 8. For each of the central components (i.e., modeling language, modeling procedure, mechanisms & algorithms), different possibilities for their specification in various degrees formalization are identified.

Within this analysis a *formalized* or a *formal definition* is defined as one that provides an

unambiguous specification that is inter-subjectively understandable and processable by different computer systems. This definition can be aligned to the one of Harel and Rumpe, stating *"'formal' is a label for any language endowed with precise and unambiguously defined syntax and semantics"* (HAREL AND RUMPE, 2004, p. 70).

Components of the Analysis Framework

Modeling languages, referring to Figure 8, are composed of *syntax*, *semantics*, and *notation*. The syntax of a modeling language is usually described in a formal way using a meta model, therefore utilizing a meta modeling approach, or a mathematical notation like FDMM (FILL ET AL., 2012) or Z (ABRIAL, 1980). A formal specification must identify the elements of the modeling language and a precise specification of the valid relationships between those elements by means of constraints and cardinalities. An informal specification of a languages' syntax is e.g., the definition of elements using natural language, e.g., "the organizational model consists of business units and relations between business units". Semi-formal specifications result from the combination of formal and informal specifications, i.e., some elements are specified formally using a meta model whereas the relations between the elements are only described generally using natural language. Many modeling languages emphasize on a formal specification of the syntax by means of a meta model, however, they lack at formally constraining the valid relationships.

Semantics and notation of a modeling language have to be investigated in more detail as they relate to both, structural and behavioral aspects of process models. Generally, a formal notation specification defines precisely the *"representation of the elements of the language"* (KÜHN, 2004), whereas a formal semantics specification assigns an unambiguous meaning, i.e., an inter-subjective understanding, to each element of the language's syntax (HÖFFERER, 2007). In the domain of enterprise modeling, Lankhorst states that *"most languages have a weak formal basis and lack a clearly defined semantics"* (LANKHORST, 2009, p. 43).

Figure 21 illustrates the analysis framework. Each analysis criteria (positioned on the right border of Figure 21) is related to the corresponding component in the hierarchical structure of modeling method components defined by KARAGIANNIS AND KÜHN (2002) (cf. Figure 8). Subsequently, each criteria is introduced briefly by discussing its specifics and pointing to techniques allowing their informal, semi-formal, and formal specification.

Notation Notation is analyzed twofold: First, *static notation* investigates a fixed notation for modeling language elements that is not changing. Second, *dynamic notation* is concerned with the question, whether a modeling language provides dynamic changes to the notation depending on the current state of the model or the current attribute value of an element. Most common modeling languages provide a static notation.

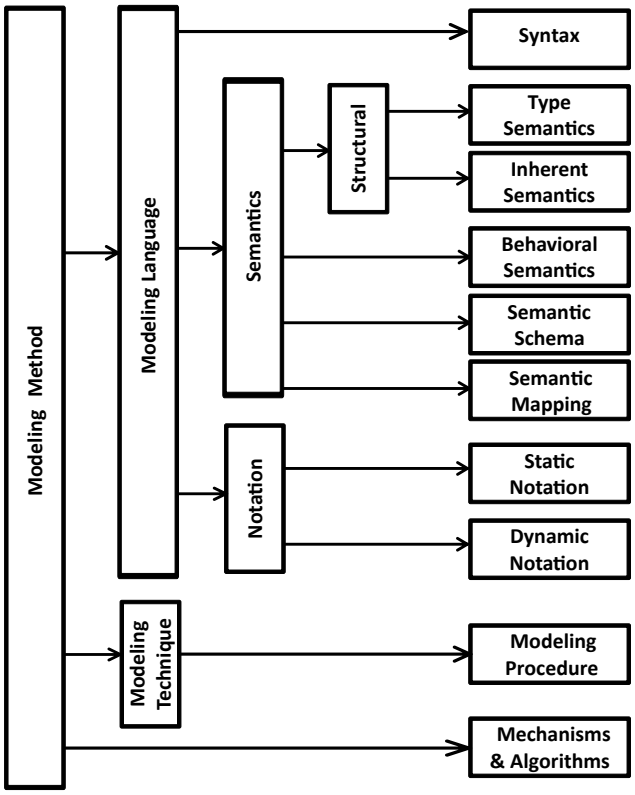


Figure 21: Analysis framework (BORK AND FILL, 2014)

The range of specifications for static and dynamic notation spans from informal by using natural language, e.g., "the element is represented by a blue car"; over semi-formal by reverting to mathematical shapes, e.g., rectangle, triangle, circle; up to formal by using a programming language or a precise mathematical description, e.g., "the element is represented by a square with an edge length of 2cm".

Considering the dynamic notation, a specification of the different states or attribute values has to be given together with a mapping function to the corresponding notation in that state. The formalization of the states and the mapping influences the formalization of the dynamic notation specification. Only if both are specified in a formal manner, a formal dynamic notation can be attested.

Semantics For semantics, the following categories have been distinguished: First, semantics are divided into *Structural Semantics*, *Behavioral Semantics*, *Semantic Domain*, and *Semantic Mapping*. In a second step, structural semantics has been further divided into *type semantics* and *inherent semantics*. The decomposition is motivated by the goal to derive the most adequate and precise analysis criteria. Behavioral semantics, type semantics, and inherent semantics are therefore investigated individually.

Structural Semantics The structural semantics of a modeling language are further decomposed into i) *Type Semantics*, usually defined with the meta model by specifying the semantics of each element (i.e., type) on the meta level; and ii) *Inherent Semantics* as introduced by HÖFFERER (2007), defining the semantics of instances of the meta model types. Figure 22 illustrates the difference between type and inherent semantics.

For the formal specification of type and inherent semantics, ontologies can play an important role. *"An ontology defines the common words and concepts (meanings) used to describe and represent an area of knowledge. [..] Ontologies include computer-usable definitions of basic concepts in the domain and relationships among them"* (OBRST, 2003, p. 366). Informal specifications can be natural language descriptions of the modeling method's domain. A semi-formal specification can be realized by linking some concepts to concepts defined in an ontology, whereas others are only specified in natural language. Often, the modeling classes are specified more formally, whereas the semantics specification for the relation classes stays on an informal level.

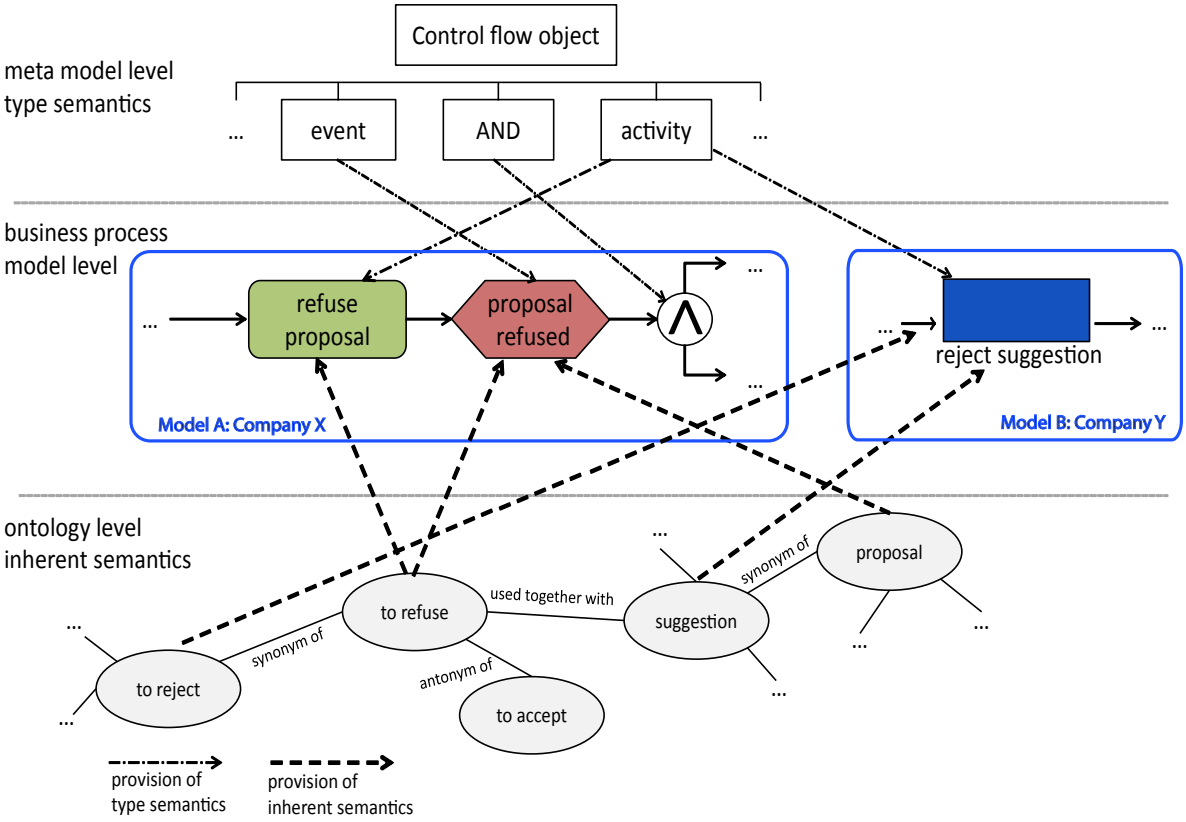


Figure 22: Type and inherent semantics (cf. (HÖFFERER, 2007, p. 1628))

Figure 22 exemplifies the relationships between type and inherent semantics. It shows excerpts of the different models and model levels together with their type and inherent semantics derivation by reverting to an event-driven process chain and a business pro-

cess model. On the meta model level and on the ontology level, both models share the same concepts, e.g., the meta concept *activity* is instantiated to *reject suggestion* and *refuse proposal*. Moreover, the attribute values, in this case the name of activities and events, are mapped to concepts on the ontology level. This integration not only improves inter-subjective understanding but also model processing by means of semantic queries or comparison of multiple process models, created according to multiple modeling languages.

Behavioral Semantics This criteria describes the degree of formalization according to the execution of the process model. A formal specification of behavioral semantics can be defined by e.g., relating the specific execution semantics to the Petri net execution semantics PETRI (1962, 1966), or by providing some algebraic specification. An informal specification can use natural language without an machine-processable, precise formula. A semi-formal specification can result in an incomplete mapping of the language's elements to the behavioral semantics specification (i.e., not all process model elements are mapped to their respective behavioral semantics).

Semantic Schema The semantic schema defines the semantic domain of the modeling language by specifying *"the very concepts that exist in the universe of discourse. As such, it serves as an abstraction of reality, capturing decisions about the kinds of things the language should express"* (HAREL AND RUMPE, 2004, p. 67). The spectrum of formalized specifications is the same as that for type semantics.

Semantic Mapping The semantic mapping defines the mapping between elements of the language's syntax and the concepts defined in the semantic schema. A formal semantic mapping relies on a formal semantic schema. Semi-formal specifications can be derived, if not all elements are associated with one exact concept of the semantic schema, whereas an informal specification of the mapping can be defined using natural language and an informal semantic schema.

As the number of investigated criteria suggests, the formalized specification of the semantics plays an important role. Ontologies provide an intuitive and efficient way to define the very concepts of a domain as well as the relationships between those concepts. However, ontologies itself comprise *"a range of models of varying degree of semantic richness and complexity"* (OBRST, 2003, p. 366). Figure 23 illustrates this range of models by ordering them in increasing semantic richness from the lower left to the upper right side (cf. (SMITH AND WELTY, 2001; DACONTA ET AL., 2003)). The spectrum ranges from weak semantics on the lower left side (e.g., Taxonomies), over conceptual models, up to strong semantics (e.g., using first order logic or modal logic).

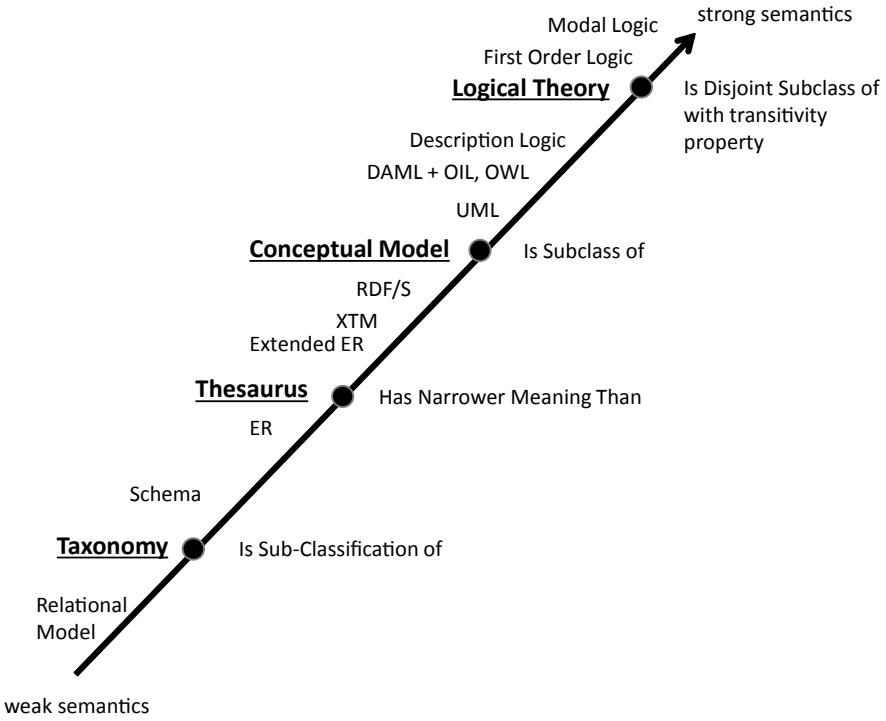


Figure 23: Ontology spectrum (OBRST, 2003, p. 367)

Modeling Procedure The modeling procedure is composed of *steps* and delivers *results* (cf. Figure 8). These two criteria describe how the user actually creates valid models, i.e., the sequence of actions performed by the modeler. A formal specification of the modeling procedure can be provided e.g., by using rule-based systems (TSALGATIDOU AND LOUCOPOULOS, 1991), Triple-Graph Grammar (TGG) (EHRIG ET AL., 2007), or constraint definition languages like Object Constraint Language (OCL) (OBJECT MANAGEMENT GROUP (OMG), 2010). TGGs *"are a rule-based technique with a formal background for specifying bidirectional model transformation[s]"* (LAUDER ET AL., 2012, p. 287).

Informal specifications can be in natural language or e.g., by a tabular specification of the different steps to be performed. A semi-formal specification can result of the combination of formal specification techniques for some modeling steps with informal specifications for some others. Another semi-formal specification can be defined by describing formally the sequence of actions that are allowed to be performed by the modeler, but staying on an informal level for the specification of the single steps itself. Therefore, a tool developer could automatically derive the valid sequence of actions to support the modeler, on the other hand it is impossible to guarantee the correct execution of the modeling actions itself.

Mechanisms & Algorithms Formal specifications for mechanisms and algorithms of arbitrary modeling methods have not been regarded in depth up to now. Whereas lot of effort has been put into providing formal approaches for the specification of syntax, semantics, and lately also notation (cf. SEIDEWITZ (2013)), the development of formal specification approaches for modeling method mechanisms and algorithms is still an open research area. Mechanisms and algorithms are e.g., simulation algorithms that can be performed on models, or model transformation algorithms, transforming a source model into a target model.

In the beginnings of enterprise modeling, the focus was on capturing the relevant aspects of an enterprise in conceptually models and linking them to each other. Recently, the focus has emerged, now including the generated models also as knowledge bases for further processing. Hence. mechanisms & algorithms are of increasing interest, too.

A formal specification in this field can be stated, if meta models are given and the model transformation is defined by specifying a meta model mapping from the source meta model to the target meta model (FILL AND KARAGIANNIS, 2013). For simulation algorithms, concrete implementations using a programming language can hold as a formalized specification. An informal specification can be achieved using natural language or use case models, whereas pseudo code notation (i.e., some programming language constructs together with natural language) can be classified as semi-formal.

Table 3 summarizes the spectrum of techniques that can be used for each analyzed criteria to produce informal, semi-formal, or formal specifications. The table provides a selection of the discussed techniques. The value *C/P* is used as an abbreviation for *Combination or Partial*. Combination means, that a semi-formal specification for a criteria can be produced by combining formal techniques with informal ones. Partial on the other hand indicates that a formal technique has been adopted, but not consequently for all parts of a criteria.

Generally, in case of the semi-formal instances, it can be often referred to one of the three cases: (1) a combination of some formal and informal specification elements (C), (2) a partly usage of a formal specification (P) (e.g., a meta model for the syntax specification but without defining the cardinalities of the relations), or (3) a dedicated semi-formal technique (e.g., pseudo code for mechanisms & algorithms).

The criteria semantic mapping cannot be described using the classes informal, semi-formal, and formal. The specification of the semantic mapping depends on the specification of the syntax on the one hand, and on the specification of the semantic schema on the other. Only if syntax and semantic schema are formally specified, a formal semantic mapping can be given. If at least one of the two aspects is on a semi-formal level (e.g., the syntax definition for TOVE), the semantic mapping remains on a semi-formal level, too.

Table 3: Formalized specification of modeling methods

Analysis criteria	Modeling Method Specification		
	Informal	**Semi-Formal**	**Formal**
Syntax	Text	C/P	Meta Model, FDMM, Z
Type Semantics	Text	Taxonomy	Ontology, Modal Logic
Inherent Semantics	Text	Taxonomy	Ontology, Modal Logic
Behavioral Semantics	Text	C/P	Petri nets
Semantic Schema	Text	Taxonomy	Ontology
Semantic Mapping	depends on syntax and semantic schema definition		
Static Notation	Text	Sketches	Mathematical, Programming Language
Dynamic Notation	Text	Sketches	Mathematical, Programming Language
Modeling Procedure	Text	C/P	TGG, OCL, BNF notation
Mechanisms & Algorithms	Text	pseudo code	Programming Language

Legend: C $\widehat{=}$ combination of formal and informal specifications, P $\widehat{=}$ partly formalized specification

4.2 Application of the Analysis Framework

The analysis framework is subsequently applied to analyze the process-related aspects of a set of enterprise modeling methods, hereby providing insights into the various degrees of formalization of the modeling method specification.

4.2.1 ARIS

The Architecture of Integrated Information Systems (ARIS) framework, first published in 1992 by SCHEER (1992); SCHEER AND SCHNEIDER (2006), introduces an integrated framework for comprehensively describing enterprises. ARIS utilizes dedicated views for the description of *functions*, *organizational structures*, *data*, *physical and non-physical output*, and a view on the *processes*. In order to describe the behavioral aspects in the process view, ARIS utilizes Event-Driven Process Chains (EPC) (HOFFMANN ET AL., 1992). EPCs are widely used in industry and still part of SAP process modeling components (KELLER AND TEUFEL, 1997). The following analysis concentrates on the process view of the ARIS framework.

Central concepts of an EPC are *function*, *event*, and *connectors* between functions and events. Functions are used to model physical and/or mental activities, transforming an input into an

output thereby fulfilling enterprise goals. Events are used to model a concrete state of the modeled system. Connectors define the control flow of the process. Logical *AND*, *OR*, and *XOR* connectors can be introduced to EPC models. These concepts are syntactically described using a meta model (cf. (GRUHN ET AL., 2008, p. 182)) and set theory (NÜTTGENS AND RUMP, 2002, p. 67ff.). Their type semantics is described informally using natural language. Moreover, the set of allowed combinations of the logical connectors and events respectively is constrained by a tabular specification of allowed combinations (KELLER ET AL., 1992, p. 14). Inherent semantics is not defined for EPCs. The behavioral semantics of EPCs is described informally. Additional research aligned the behavioral semantics to Petri net theory (CHEN AND SCHEER, 1994). Therefore, the behavioral semantics can be stated as formal. EPCs utilize the simulation of its process instances. The static notation of function, event, and connector is semi-formally defined by an informal natural text description and a legend, illustrating sample shapes, i.e., a rounded rectangle for functions, a hexagon for events, and edges and arrows for connectors. A dynamic notation is not defined.

A considerable effort has been put on a more comprehensive formal specification of the semantics of EPCs (e.g., NÜTTGENS AND RUMP (2002)), especially considering the non-local semantics (e.g., KINDLER (2004)). The authors argue that the formal aspects of the Petri net semantics are not sufficient to describe the behavioral semantics of EPCs appropriately. In order to overcome that shortcoming, several approaches have been published that introduce a comprehensive formal specification of both, behavioral semantics and type semantics. Other authors even argue, that *"there is no sound formal semantics for EPCs that is fully compliant with the informal semantics"* (AALST ET AL., 2002, p. 71), however providing their own formalization some years later (MENDLING AND AALST, 2007).

Semantic domain and semantic mapping are defined on an informal level. As the initial publication of EPCs defines some decomposition guidelines for functions and events, an informal specification of the modeling procedure is attested. Table 4 illustrates the results of the analysis framework applied to the process-related part of the ARIS framework.

Table 4: Formalization of ARIS

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
ARIS	●	○	n/a	●	○	○	◐	n/a	○	●

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.2 BPMS

The Business Process Management Systems (BPMS) paradigm, which integrates *"the organizational, analytic as well as the IT aspects of business processes, is an approach for the management of business processes"* (KARAGIANNIS ET AL., 1996, p. 86). The method has been developed at the University of Vienna by a team led by Prof. Karagiannis. Besides the theoretical concepts, also a commercial modeling tool for the BPMS method has been developed, called Adonis. The tool is still being used in education and a wide range of industrial projects (HARMON, 2010).

BPMS integrates three abstraction levels: *business level*, *execution level*, and *evaluation level*. Each level is defined by one or more dedicated *BPMS-Processes*: *strategic decision process*, *re-engineering process*, *resource allocation process*, *workflow management process*, and *performance evaluation process*. Modeling and analysis of business process models is considered in the re-engineering process. Therefore the following analysis concentrates on this part of the BPMS method.

The syntax of the business process modeling component is described formally using an algebraic notation (KARAGIANNIS ET AL., 1996, p. 88ff.). The type semantics of the central modeling elements, activities, subprocesses, and control flow, is described informally using natural language, i.e., *"Activities are the atomic units of a process, e.g., the working units which cannot or should not be divided any more"* (KARAGIANNIS ET AL., 1996, p. 88). Sample simulation algorithms are informally defined by HERBST ET AL. (1997) and implemented in the Adonis BPMS modeling tool[19]. Some algorithms, i.e., querying of BPMS models with the BPMS tool, have recently been specified semi-formally (FILL AND KARAGIANNIS, 2013). HERBST (2001) introduced a mapping from BPMS process model elements to the concepts of Petri nets, therefore providing a formal semantic mapping and a formal semantic domain. The behavioral semantics is also inherited through the Petri net semantics.

Although the method defines dependencies between the several BPMS processes, no modeling procedure, in the sense of a sequence of modeling steps, for the creation of business process models is defined. The static notation of BPMS business process models is defined semi-formally by providing graphical visualizations, i.e., samples, for some elements of the method, but not for the relations (HERBST, 2001). A dynamic notation is not defined.

Table 5 visualizes the analysis results for the BPMS modeling method.

[19] Adonis BPMS modeling is part of the free ADONIS Community Edition modeling tool developed and distributed by the BOC Group, `http://www.adonis-community.com/`, last checked: 2013-11-01

Table 5: Formalization of BPMS

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
BPMS	●	○	n/a	●	●	●	◐	n/a	n/a	◐

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.3 HORUS

Horus is an enterprise modeling method that focuses on business process engineering. The method includes steps for integrated modeling of business processes, improvement of business processes, and application of the created models (SCHÖNTHALER ET AL., 2012). For these purposes, the Horus method comprises four phases: 1) preparing process optimization projects, 2) elaborating the strategy and architecture, 3) analyzing business processes, and 4) applying the results. To investigate the formalization of the process-related aspects in Horus, a restriction to the third phase of Horus is imposed. The core of this phase are so-called *procedure models* that can be further linked to organizational models, rule models, object models, key figure models, resource models, and risk models. The procedure models are based on high-level Petri nets, which are extended to a Horus specific variant, denoted as *XML nets*. *"XML nets are a formal, graphical modeling language that allows to model both the flow of XML documents and the control flow of the underlying business process"* (LENZ AND OBERWEIS, 2003, p. 244). In XML nets, the objects in the places are XML documents and transitions are operations on these XML documents using XQuery[20] statements.

For the syntax of Horus procedure models a mathematical specification is available based on the FDMM formalism (FILL ET AL., 2013). Regarding the type semantics of procedure models, the underlying Petri nets together with formal descriptions of XML nets provided by Lenz and Oberweis through formal mappings to XML and XQuery specifications (LENZ AND OBERWEIS, 2003) can be characterized as formal. Accordingly, also the semantic mappings and the semantic domain for procedure models are formally defined.

The behavioral semantics of procedure models are also formally defined by utilizing the behavioral semantics of Petri nets. The static notation of procedure models is described semi-formally through graphical illustrations, i.e., samples. In addition, procedure models also fea-

[20] See the XQuery specification for further details: `http://www.w3.org/TR/xquery/`, last checked: 2013-11-01

ture a dynamic notation, e.g., for depicting organizational resources that are linked to transitions (LENZ AND OBERWEIS, 2003, p. 54ff.). These dynamic aspects are also illustrated semi-formally.

For the modeling procedure, SCHÖNTHALER ET AL. (2012) showed detailed diagrams using a Petri-net-like notation to illustrate the utilization of the various model types. However, this is not detailed down to the level of modeling objects. Therefore, the modeling procedure is classified as semi-formal. Horus procedure models can be simulated. These algorithms are explained by SCHÖNTHALER ET AL. (2012) in a semi-formal style with mainly textually describing their behavior together with examples for illustrating corresponding calculations.

The results of the analysis are summarized in Table 6.

Table 6: Formalization of HORUS

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
HORUS	●	●	n/a	●	●	●	◐	◐	◐	◐

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.4 IDEF

In the 1980s, the Air Force Information Integration for Concurrent Engineering (IICE) program started to work on the *IDEF3 process modeling language*. IDEF3 is part of the IDEF family of languages, a comprehensive set of modeling methods initialized by the United States military with the goal of generating more efficiency through information systems modeling. *"IDEF3 descriptions are developed from two different perspectives: process-centered and object-centered"* (MAYER ET AL., 1995, p. 21). The following analysis concentrates on the process modeling related perspective of IDEF3.

Central model elements of the IDEF3 modeling language are *Unit of Behavior (UOB)* boxes, temporal relations (i.e., *Simple Precedence Links*, *Constraint Precedence Links*, and *Relational Links*), and *Junctions*. Junctions enable the specification of logical connections for precedence links (i.e., *AND*, *OR*, *XOR*). Additionally to these quite common logical connectors, the syntax of the IDEF3 modeling method also provides *Sync AND* and *Sync OR* connector types, enabling more precision while modeling the temporal aspects of business process executions.

Providing these specialized connector types immediately leads to one of the application scenarios of IDEF3 models: simulation. Finally, the specification also includes *Referents* concepts to enable a more efficient application of the method. The aim of the Referent concept is to decrease inconsistency and unnecessary modeling actions by providing the modeler the ability to reference or decomposed already defined IDEF3 process models.

The syntax of IDEF process models is described semi-formally by a legend of its elements without defined cardinalities, appended with an informal description using natural language (cf. (MAYER ET AL., 1995, p. 22)). The type semantics is specified informally, i.e., using natural language to describe the concepts of the method. An inherent semantics is not specified for IDEF3.

The behavioral semantics of IDEF3 models are not formally or mathematically described. However, the method provides a structural approach towards the specification of the behavioral semantics of the process models. This approach is based on semi-formally specifying the probabilities of different process paths and the semantics of forking and joining process models. However, IDEF3 models can not be simulated directly, they must be transformed into corresponding simulation models, enriched with simulation-tool specific (e.g., PROSIM[21]) formalized attributes. Considering semantic mapping and semantic schema, IDEF3 stays on an informal level.

The static notation of the process modeling method's elements is semi-formally defined by a legend, consisting of sample visualizations of all relevant modeling language elements. Moreover, a rich set of sample models is given, showing concrete IDEF3 models (cf. (MAYER ET AL., 1995, p. 22ff.)). The process related modeling method does not provide a notation that changes dynamically. However, the sample models show, that the current state of the object states can be customized by individual labels, represented in the notation. Moreover, UOB symbols comprise a *Node Ref* and a *IDEF Ref* in their graphical visualization. Hence, a informal specification of the dynamic notation can be attested.

IDEF3 provides a rich set of heuristic modeling knowledge. Moreover, MAYER ET AL. (1995) propose a structured way of developing IDEF3 process descriptions, informally describing a sequence of the following steps: 1) *Collect*, collect all information about the processes to be modeled; 2) *Classify*, classify the constituents of the multiple processes, e.g., the object types, object states, and relations ; 3) *Organize*, organize the collected information by means of IDEF3 structures; 4) *Validate*, compare the created IDEF3 structures syntactically with the IDEF3 syntax, and semantically with the collected information about the real life phenomenon; 5) *Refine*, adjust the IDEF3 representation according to the discussions and the new perspective on the phenomenon (MAYER ET AL., 1995, p. 96f.). Each of this steps is supported with con-

[21] PROSIM homepage: `http://www.kbsi.com/technologies/prosim`, last checked: 2015-04-08

crete guidelines, tools, and sequence diagrams. The method also suggests some dependencies between the sequence of steps of creating models, i.e., first the *Process Schematic* and then the *Transition Schematic* model should be created. Hence, the modeling procedure of IDEF3 is very comprehensively described, but overall on a semi-formal level.

The IDEF3 method comprises the *Elaboration Language*, a formal language enabling constraining and processing of process models. This language is a powerful weapon when it comes to the specification of constraints and/or the definition of the behavioral aspects of the process models. For any *kind* element, pre- and post-conditions can be defined based on the object state. Moreover, the method provides the possibility to attach complex state transition logics, e.g., by referring to transition probabilities. Hence, some mechanisms & algorithms are defined for IDEF3 in a formalized manner.

Table 7 summarizes the results of the IDEF3 analysis.

Table 7: Formalization of IDEF3

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
IDEF	◐	○	n/a	◐	○	○	◐	○	◐	●

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.5 MEMO

First publications on the Multi-Perspective Enterprise Modeling (MEMO) method have been published in the early 1990s (FRANK, 1994). Since then the method has been under further study and development (FRANK, 1999, 2002, 2010, 2012). Consequently, several extensions of MEMO, i.e., on internal control modeling (CONTROLML (HEISE ET AL., 2014)), on performance measurement modeling (METRICM (STRECKER ET AL., 2012)), on the modeling of IT risk assessment (RISKM (STRECKER ET AL., 2011)), or on organization modeling and model-driven software development (MEMOCENTERNG (GULDEN AND FRANK, 2010)) have been developed.

The specifications for MEMO OrgML are divided into two research reports, one focusing on the organizational structure (cf. (FRANK, 2011a)), the other on business processes (cf. (FRANK, 2011b)). The following analysis therefore concentrates on the latter part of the specification.

MEMO OrgML *"provides concepts to model a business process in a detailed way"* (FRANK, 2002, p. 78). The aim for detailed business process modeling requires a rich set of modeling language elements. These elements are syntactically specified in a formal way using a comprehensive meta model (FRANK, 2011b, p. 56). All concepts together with the semantics of their interrelations are subsequently explained in an informal way (cf. (FRANK, 2011b, p. 57ff.)). The meta model is complemented with formal constraints. These constraints serve either for *"preventing particular edit operations immediately"* (FRANK, 2011b, p. 57), referred to as *ad hoc constraints*, or enable checking the model integrity on demand, referred to as *on demand constraints*. The syntax of MEMO OrgML can therefore be characterized as formal.

The graphical notation of the elements is specified in a semi-formal way by presenting samples in addition to an informal description, i.e., a time interval event as a triangle combined with a clock, or a manual subprocess by using the notation of a suprocess and adding a worker with a desk in front of it. As the authors emphasize on the notation, they provide two variants for representation, a *matt* and a *glossy* one. The same specification technique is used for the associations. As the notation changes based on the current attribute value of e.g., a subprocess, i.e., changing the attribute value of the attribute *type* from *manual* to *auto* replaces the worker and his/her working desk with a personal computer that is placed in front of the subprocess notation. Thy dynamic notation is therefore defined on a semi-formal level.

As for any other current extension of MEMO, all elements of the extended modeling languages are syntactically and semantically aligned to those defined in the meta meta model of MEMO (FRANK, 2010). Therefore, the type semantics of MEMO OrgML can be stated as semi-formal. It cannot be stated as formal because the type semantics of the meta meta model elements is only on an informal level. Due to the constraints defined on the meta meta model level and the integration of different modeling languages on the meta level, i.e., OrgML, SML, and OML, following an agreement on the semantics, the type semantics of OrgML can be stated semi-formal. An inherent semantics for the concepts of the organization modeling language is not defined. The behavioral semantics of OrgML models are discussed extensively on an informal level (cf. (FRANK, 2011b, p. 18ff.)). The discussion covers the abstract syntax and the semantics for basic control structures, e.g., sequence, branching, concurrency, synchronization; and advanced control structures, e.g., arbitrary sequences, arbitrary sequences with partial order, synchronization exceptions, variable number of concurrent instances. The execution of the process models is not targeted by the specification. However, *"it should be sufficient for mapping respective business process models to representations that are executable"* (FRANK, 2011b, p. 2).

The process context defines the interrelations between concepts defined in the OrgML meta model and the meta models of other parts of MEMO, e.g., *OrganizationalUnit* from the meta model of organizational structures (FRANK, 2011a, p. 50). The relations between those two

meta models are precisely defined using the specific concepts of each meta model. The meta model mapping also defines the cardinalities for those relations, i.e., from *ControlFlowSubprocess* to *OrganizationalUnit*. The relation defines, that each ControlFlowSubprocess instance is related to zero or one (0,1) OrganizationalUnits, and on the other hand, that each OrganizationalUnit instance is *in_charge_of* zero to many (0,*) ControlFlowSubprocess instances. The semantic schema can therefore be stated as informal, because the attribute values of the instances can be related to instances of other MEMO models. The semantic mapping, based on the semantic schema and defined by concrete relations with precisely defined semantics and cardinalities, is also on an informal level.

The comprehensive specification for MEMO OrgML does not include a concrete modeling procedure. Therefore, it is characterized as not available. However, the specification provides a set of examples, illustrating the application of the modeling method (FRANK, 2011b, p. 89ff.). The examples include several hints, i.e., heuristic modeling knowledge gained during previous applications, that might ease the application of OrgML: First, create a *business process map* utilizing a *""ballpark view" of a company's business process types"* (FRANK, 2011b, p. 89); Second, create a *process (de-)composition diagram* illustrating *"function composition of processes"* (FRANK, 2011b, p. 90); Third, create a *process inheritance diagram "representing specialisation relationships between subprocess types"* (FRANK, 2011b, p. 90).

The Memo OrgML specification does not particularly define mechanisms and algorithms. However, the effort made to provide the rich set of e.g., different events, notifications, exceptions, together with the possibility of integrating e.g., organizational units, enables OrgML models to be analyzed and interpreted in manifold ways. These analysis and query functionality is not expatiated. Some basic simulation algorithms (cf. (FRANK, 1992)) and analysis queries (cf. (FRANK, 1994)) are exemplified. Mechanisms and algorithms can therefore be characterized as semi-formal. Table 8 summarizes the results of the analysis.

Table 8: Formalization of MEMO

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
MEMO	●	◐	n/a	○	○	○	◐	◐	n/a	◐

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.6 SOM

The Semantic Object Model (SOM) (FERSTL AND SINZ, 2013) is a multi-perspective enterprise modeling method developed by Ferstl and Sinz at the University of Bamberg. SOM compromises a layered approach for a comprehensive specification of an enterprise with an *enterprise plan* on the top layer, a *business process model* on the central layer, and a models for the *specification of resources* on the bottom layer. *"Each layer describes the business system as a whole, but with respect to a specific perspective on the model"* (FERSTL AND SINZ, 2006, p. 347).

An emphasis of SOM is on the specification of business process models using a multi-view approach (BORK AND SINZ, 2013). SOM business process models integrate behavioral and structural aspects, modeled in different views, into one comprehensive business process model. The view for structural aspects is called *Interaction Schema*, the view depicting the behavioral aspects is called *Task-Event Schema*. Additionally, SOM provides views on the decomposition of business objects and business transactions, in the following referred to as *Object Decomposition Schema* and *Transaction Decomposition Schema*, respectively. On the top level and bottom level additional views are defined. However, the following analysis concentrates on the business process modeling part of the SOM method. For a complete and detailed introduction to the SOM method see section 7.1.

The syntax of SOM business process models is described using a meta model. This meta model also includes the static notation of the languages' elements by providing illustrations of their respective shapes, e.g., environmental objects as ellipses, business objects as rectangles, business transactions as directed arrows (FERSTL AND SINZ, 2013, p. 221). A dynamic notation for SOM is not defined. The type semantics of the business process model elements is described informally using natural language, e.g., *"A **business transaction** [..] transmits a **good or service** to a customer business process or receives a good or service from a supplier business process"* (FERSTL AND SINZ, 2006, p. 352), but referring to the concepts of systems theory, transaction-based coordination, and object-orientation. Inherent semantics are not specified.

The behavioral semantics for SOM process models is inherited from Petri nets and extended with e.g., the concepts of pre- and post-conditions, and the relation of transitions to business objects. Semantic mapping as well as semantic domain are described informally.

The modeling procedure for SOM business process modeling is described with precise decomposition rules that can be applied to business objects and business transactions. SOM also defines the initial scheme of a business process model. *"At the initial level of a business process model, a business object [..] produces goods and services and delivers them to customer business processes"* (FERSTL AND SINZ, 2006, p. 352)). The decomposition rules are specified using a Backus-Naur notation (see Table 17 for a complete list of the decomposition rules). They specify how modelers can apply them recursively, thereby detailing and refining the initial

business process model. In addition, the application of the decomposition rules reveals the coordination of the involved business objects. Modeling mechanisms and algorithms, considering the business process modeling part of SOM, are not defined.

The results of the analysis are summarized in Table 9.

Table 9: Formalization of SOM

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
SOM	●	○	n/a	●	○	○	◐	n/a	●	n/a

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.7 TOVE

The diversity of enterprise models and legacy systems supporting the creation of those models has led to the *correspondence problem* (FOX AND GRUNINGER, 1998). The correspondence problem covers the problem of comparing different enterprise models. *"Although each enterprise model might represent the same concept, for example, activity, they will have a different name, for example, operation versus task"* (FOX AND GRUNINGER, 1998, p. 110). The authors therefore introduced the idea of a General Enterprise Model (GEM), that can be extended to a concrete domain, resulting in the specification of a Deductive Enterprise Model (DEM). GEMs consist of three parts: 1) a taxonomy of object classes; 2) for each object class, relations to other object classes in addition to a definition of the semantics of the relation, and 3) a set of attributes for each object class, together with the intended meaning of the attributes (FOX AND GRUNINGER, 1998). Fox et al. used this GEM to develop the Toronto Virtual Enterprise (TOVE). TOVE has it's name due to its origin, the University of Toronto.

In contrast to the most common enterprise modeling methods, TOVE does not only support modeling of enterprises using a process-oriented model. Moreover, the goal of TOVE is to provide a common knowledge base. This knowledge base then allows the deductive answering of queries like: *"Given some initial state, which resources are perishable?"* or *"Given some scenario of events, when will a particular resource spoil?"* (FOX AND GRUNINGER, 1998, p. 119f.).

The goal of the TOVE project is to create an ontology of an enterprise, defining the semantics of each component, implementing a deductive approach by transforming the semantics into

axioms. These axioms enable automatic, deductive answering of common questions on the enterprise. Graphical symbols for these components are also defined within the project (FOX, 1992). The following analysis concentrates on the *Activity-State Model* which is based on the *Activity-State Ontology* of TOVE.

The syntax of the Activity-State model can be classified as semi-formal, because on the one hand, all elements of the model are described comprehensively using taxonomies, but on the other hand, a formal specification according to the allowed relations between these elements is missing (cf. (FOX ET AL., 1993)).

Structural semantics and behavioral semantics are both specified formally. This not only holds for the type semantics (as part of the structural semantics) but also for the inherent semantics. Both are specified using ontologies. The activity-state-time ontology uses the situation calculus (REITER, 1991) as foundational theory for describing the semantics of the ontology of the concepts activity, state, and time (FOX AND GRUNINGER, 1998). Therefore, the behavioral semantics is completely specified formally.

The static notation of the elements are defined semi-formally by providing some sample models, including sample elements of Activity-State models (cf. (FOX AND GRUNINGER, 1998, p. 118), (FOX ET AL., 1993, p. 427), visualizing activities as ovals, states as rectangles, relations between these concepts as directed arrows with a solid line). A dynamic notation is not defined. The semantic domain of Activity-State models is formally defined using ontologies, specifying the semantics of the domain's concepts and the relations between these concepts. The semantic mapping however is on a semi-formal level as is relies on the semi-formal syntax.

Considering the Activity-State model exclusively, no modeling procedure is defined. As the model is directed towards a deductive competence, the questions - defined in first-order logic and implemented in the programming language Prolog - represent some mechanisms & algorithms of TOVE in a formal way. The results of the analysis of the TOVE approach are summarized in Table 10.

Table 10: Formalization of TOVE

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
TOVE	◐	●	●	●	●	◐	◐	n/a	n/a	●

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

4.2.8 UML

The Unified Modeling Language (UML) is the de-facto industry standard for the object-oriented specification of software-intense systems (ENGELS ET AL., 2001). *"The objective of UML is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes"* (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 1). The first versions of the UML, released in the late 1990s and early 2000s by Booch, Jacobsen, and Rumbaugh, *"originated with three leading object-oriented methods (Booch (*BOOCH*, 1982), Object-Modeling Technique (OMT) (*RUMBAUGH ET AL.*, 1990), and Object-Oriented Software Engineering (OOSE) (*JACOBSON*, 1992)), and incorporated a number of best practices from modeling language design, object-oriented programming, and architectural description languages"* (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 1).

Over the years, the standard has emerged steadily. Not only regarding the number of languages and application domains, but also in precision according to the specification of the languages. The current version, UML 2.4.1, provides a rich set of diagrams for different aspects of a system (e.g., *class diagrams* or *component diagrams* for capturing structural aspects, and *activity diagrams* or *sequence diagrams* for behavioral aspects). UML is used on a broad basis today. Besides the set of diagrams, the current version also introduces the possibility to create *UML profiles*. These UML profiles enable the UML to be applicable in new and emergent domains. Nevertheless, as the analysis concentrates on behavioral aspects, an investigation of UML *Activity Diagrams* is undertaken in the following.

Activity Diagrams (ADs) are used to specify the dynamic behavior of the modeled system by defining *activities* and *relations* between these activities. The central elements of ADs, i.e., activities, activity edges, are formally specified using a meta model (OBJECT MANAGEMENT GROUP (OMG), 2011c, p. 14f.) derived from the MetaObject Facility (MOF) meta meta model (OBJECT MANAGEMENT GROUP (OMG), 2011b). Activity diagrams utilize Petri net like semantics. Therefore, the behavioral semantics of ADs can be classified as formal. Concerning the structural semantics, the UML provides a semi-formal type semantics for the elements of ADs. Besides the informal description of the semantics using natural language, each of the language's elements is related to a concept of the meta model, and therefore to a concept of the MOF, by a generalization relationship. Inherent semantics is not specified.

The static notation of each element is defined semi-formally by describing textual and graphical notation of the elements using natural language and example diagrams, respectively. A dynamic notation is not defined by the UML.

According to semantic mapping and semantic domain, no specifications have been defined. Although the integrated MOF meta meta model allows for hands-on definition of meta model mappings for the transformation of ADs into related diagrams, no mechanisms and algorithms,

specifically for activity diagrams, are defined. The same holds for the modeling procedure, as the UML generally doesn't define any procedure for its application. Table 11 summarizes the results of the performed analysis.

Table 11: Formalization of UML

	Syntax	Type Semantics	Inherent Semantics	Behavioral Semantics	Semantic Schema	Semantic Mapping	Static Notation	Dynamic Notation	Modeling Procedure	Mechanisms & Algorithms
UML	●	◐	n/a	●	○	○	◐	n/a	n/a	n/a

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

Current Developments on the Formalization of the UML

The preceding analysis evaluated the current specification version 2.4.1 of the UML, especially the UML activity diagram specification. However, there are currently very interesting research efforts aligned to the UML and the formalization of UML specifications. ENGELS ET AL. (2001) introduced a semantic domain and a semantic mapping in order to utilize concrete consistency tests between different UML diagrams. Others introduced a formal semantics by mapping the concepts of *SysML Activity Diagrams* (OBJECT MANAGEMENT GROUP (OMG), 2012b), an UML activity diagram profile directed towards systems engineering applications, into a mathematical expression called *Activity Calculus* (JARRAYA ET AL., 2009; JARRAYA AND DEBBABI, 2012).

The Object Management Group (OMG) itself is also working on the introduction of formalized specifications, e.g., for the definition of graphical notation specifications. The UML is working on the *Diagram Definition (DD)* (OBJECT MANAGEMENT GROUP (OMG), 2012a) language. Using the DD, one is able to define the graphical notation of diagram elements in a formal way by providing a mapping between the abstract syntax of the language and the provided diagram graphics of the DD. *"The Diagram Definition (DD) specification provides a basis for modeling and interchanging graphical notations, specifically node and arc style diagrams as found in UML, SysML, and BPMN, for example, where the notations are tied to abstract language syntaxes defined with MOF"* (OBJECT MANAGEMENT GROUP (OMG), 2012a, p. 1).

Another interesting development direction around the UML is the *Foundational UML (fUML)* (OBJECT MANAGEMENT GROUP (OMG), 2013). With the efforts for fUML the OMG defines

an execution semantics for a subset of the UML concepts. The goal is to enable *"compliant models to be transformed into various executable forms for verification, integration, and deployment"* (OBJECT MANAGEMENT GROUP (OMG), 2013, p. 1). Accordingly, conformance of models can be evaluated on two aspects OBJECT MANAGEMENT GROUP (OMG) (2013): 1) *Syntactic Conformance:* A conforming model must be restricted to the abstract syntax subset defined for the fUML, and 2) *Semantic Conformance:* A conforming execution tool must provide execution semantics for a conforming model, similar to the semantics, specified for fUML.

Summarizing the current developments, it should be stated, that there are significant efforts to introduce more formalized aspects into the specification of several parts of the UML family of languages. However, because only current developments based on the UML 2.0 were presented, it should be noted, that formalization of the former UML standards have also been considered in the past (cf. (BREU ET AL., 1997; WANG ET AL., 1997; FRANCE ET AL., 1998; RICHTERS AND GOGOLLA, 1998)). EVERMANN AND WAND (2001) introduced an ontology-approach for enabling a formal semantics for UML models. McUmber and Cheng *"introduce a general framework for formalizing a subset of UML diagrams in terms of different formal languages based on a homomorphic mapping between metamodels describing UML and the formal language"* (MCUMBER AND CHENG, 2001, p. 433).

4.3 Discussion

The results of the analysis are summarized in Table 12. The table shows, that the investigated enterprise modeling methods differ significantly according to their degree and number of formalized specifications. The results show that, based on the presented analysis framework, neither one method is completely specified formally, nor completely informally.

TOVE is the only method that provides a formally specified inherent semantics. *HORUS* and *MEMO* are the only methods providing at least a semi-formal specification for the dynamic notation criteria. With the increasing possibilities of modeling tool development platforms, it is assumable, that in the future several methods will also incorporate a formalized dynamic notation specification.

Having five out of the ten analysis criteria defined formally and four more criteria defined on a semi-formal level, HORUS is obviously the method with the highest degree of formalization in the analysis. The method with the second highest degree of formalization is TOVE. TOVE also has five criteria formally specified and three more are specified on a semi-formal level.

On the other side of the spectrum is the UML, having in the current version only two criteria specified on a formal level and two more specified on a semi-formal level. Moreover, the UML is the method with the most appearances of the value n/a, meaning that no information consid-

ering an analysis criteria is defined. The analysis results for the UML indicate the motivation of the Object Management Group (OMG) to introduce more formalized specifications for several parts of the standard. These extensions (cf. the preceding paragraph), most of them are still under development, will enable the UML to be specified in a more formalized way.

Table 12: Overview of the analysis results

	ARIS	BPMS	HORUS	IDEF3	MEMO	SOM	TOVE	UML
Syntax	●	●	●	◐	●	●	◐	●
Semantics								
Structural Semantics								
Type Semantics	○	○	●	○	◐	○	●	◐
Inherent Semantics	n/a	n/a	n/a	n/a	n/a	n/a	●	n/a
Behavioral Semantics	●	●	●	◐	○	●	●	●
Semantic Schema	○	●	●	○	○	○	●	○
Semantic Mapping	○	●	●	○	○	○	◐	○
Notation								
Static Notation	◐	◐	◐	◐	◐	◐	◐	◐
Dynamic Notation	n/a	n/a	◐	○	◐	n/a	n/a	n/a
Modeling Procedure	○	n/a	◐	◐	n/a	●	n/a	n/a
Mechanisms & Algorithms	●	◐	◐	●	◐	n/a	●	n/a

Legend: n/a $\widehat{=}$ not available, ○ $\widehat{=}$ informal, ◐ $\widehat{=}$ semi-formal, ● $\widehat{=}$ formal

Introducing formalization enables the resulting models to be inter-subjectively understandable (i.e., unambiguous) and processable by different computer systems (e.g., for analysis or simulation). Automated verification, validation, and model testing do also significantly benefit from the availability of formal specifications. Generally, introducing a formal specification enables testing model conformity on a deeper level.

Whereas most enterprise modeling methods allow for syntactic conformity, i.e., checking whether the elements in the model conform to the abstract syntax defined in the meta model, some others already provide semantic conformity checking mechanisms, i.e., checking whether the semantics of a model element is consistent. Accordingly, modeling tools can be classified into:

1. Tools, providing syntactic conformity, and

2. tools, providing syntactic and semantic conformity.

Models created with formally specified semantics can be compared on a semantic basis because the subjectivity of modeling is limited. This fosters an inter-subjective understanding of the models. Moreover, reasoning and deduction based on models is enabled. TOVE is the only method in the analysis providing a completely formal specification of its semantics, i.e., type semantics, inherent semantics, and behavioral semantics. The created TOVE models therefore enable comparison on a semantic basis. For utilization in enterprises or in a collaborative scenario, it is also beneficial to have not only a generally agreed syntax but also semantics.

The formal specification of the behavioral semantics enables the seamless interchange e.g., between a modeling tool and a simulation tool, e.g., for BPMS models in the ADONIS modeling toolkit. The analysis shows, that in the field of process-related modeling, Petri net semantics are the common denominator. A large number of modeling methods rely their behavioral semantics on the semantics of Petri nets. Hereby, re-use of existing analysis functionality e.g., simulation approaches for Petri nets is enabled.

In the case of the SOM modeling method, the modeling procedure is specified completely formalized. This enables to check a modeling tool according to its conformity to the procedure of creating valid SOM models. Obviously, this decreases the level of freedom for the tool developer and the modeler, but it should foster the correct utilization of the method.

A formalized specification of mechanisms & algorithms of a modeling method enables the automatic verification and validation of an implementation, independent from a certain development platform or programming language. Hence, tool developers can be supported with precise requirements for the tool's functionality. The discussion shows, that the level of inter-operability depends significantly on the level of formalization in the specification. To an extreme, applying the diagram definition approach of the UML (OBJECT MANAGEMENT GROUP (OMG), 2012a), it is possible to provide conformity not only on syntactic and semantic level, but also on a notational level. Agreeing on the notation would enable human beings to interpret models, created with different modeling tools, more efficiently.

Besides these positive effects it must be stated, that introducing formalized specifications also reduces the degree of freedom for modelers and obviously *"increases the effort involved in creating the specifications"* (MESEGUER AND PREECE, 1996, p. 317). As modeling is generally a very creative and subjective task, it may in some cases be counterproductive to e.g., formalize the modeling procedure. The desirable degree of formalization must therefore be decided for each component of a modeling method and the intended usage of the modeling method individually.

Current activities of the UML, e.g., fUML (OBJECT MANAGEMENT GROUP (OMG), 2013),

diagram definition (OBJECT MANAGEMENT GROUP (OMG), 2012a), on the other hand indicate, that more aspects of modeling methods should be specified in a formalized way. *"However, to enable meaningful exchange of model information between tools, agreement on semantics and notation is required"* (OBJECT MANAGEMENT GROUP (OMG), 2011c, p. 1). Like Bertrand Meyer already stated in 1985 for the usage of formal specifications, the intention of this analysis was not to *"advocate formal specifications as a replacement for natural-language requirements; rather, we view them as a complement to natural-language descriptions and [..] ad an aid in improving the quality of natural-language specifications"* (MEYER, 1985, p. 6). The analysis, and specifically the discussion of the results, showed how formalized specifications can reduce misunderstanding and uncertainty on the one hand and foster a consistent interpretation and machine processing on the other.

Meyer also discussed the question *"Once we have a formal specification, what can we do with it?"* (MEYER, 1985, p. 20). He proposed three possible applications scenarios for the formal specification: (1) *"Relying on the specification as a basis for the next stages of the software life cycle - program design and implementation"* (MEYER, 1985, p. 20), as the most intuitive thing to do with a specification, or (2) Using the formal specification *"as a starting point for better natural-language requirements"* (MEYER, 1985, p. 20), i.e., using the formal specification to revise or generate a specification in natural language, and (3) Using the formal specification as a knowledge base by *"querying the specification to learn as much as possible about properties of the problem and valid solutions"* (MEYER, 1985, p. 20). The latter two arguments follow the initial statement, that the need for formal specifications does not compel the replacement of a natural language specification. However, they *"help expose ambiguities and contradictions because they force the specifier to describe features of the problem precisely and rigorously"* (MEYER, 1985, p. 22). This attitude on formal and formalized specifications, enriching natural language specifications, is adopted by the methodical support for the development of multi-view modeling tools, the MUVIEMOT method, in the following.

4.4 Summary

The preceding section first described a spectrum of formalization possibilities for the different components of modeling methods. These components have been further decomposed in order to define a comprehensive analysis framework. This framework has been applied to a set of enterprise modeling methods, investigating on the degree of formalization considering their process-related aspects. A comprehensive discussion of the benefits and drawbacks of formalization followed by a short outlook into current and possible future directions in the field of formalized modeling method specification concluded this part of the thesis.

5 Conceptual Modeling Using Integrated Multiple Views

> *At this point, it is still largely up to the modeler to construct different views of the same system. How best to harness a modeling system to assist the user with this task still seems to be a largely open problem.* (BROOKS ET AL., 2008, p. 13)
>
> *(Christopher Brooks et al.)*

In order to analyze multi-view modeling methods and to design a development method for the conceptual design of multi-view modeling tools, the specific characteristics of multi-view modeling methods need to be identified. These characteristics establish requirements for a modeling method aiming at supporting the conceptual design of multi-view modeling tools.

This section is structured as follows: First, section 5.1 provides definitions of the focal points, the succeeding sections build upon. Hence, a common understanding, of the terms *view*, *viewpoint*, and *multi-view modeling* is established. Subsequently, determinants of viewpoints are investigated in section 5.2, before section 5.3 describes a dichotomy of integrated multi-view modeling approaches, referring to the relationships between meta models and viewpoints. An emphasis of this section is on multi-view modeling operations and the principles of carrying out multi-view modeling in section 5.4.

5.1 Integrated Multi-View Modeling: A Definition

Section 2.2.2 discussed a set of definitions for the terms view, viewpoint, and multi-view modeling found in the related literature. In order to provide a sound theoretical foundation for this thesis, the understanding of these concepts in this thesis is now introduced.

Definition 1 (Viewpoint) *A viewpoint defines a certain perspective on a system under study on an abstract level. The specification of this perspective comprises a modeling language (composed of syntax, semantics, and notation), specifying the available concepts; a set of stakeholders, addressing the aim of the viewpoint; and a metaphor, delimiting the relevant subarea of the real world to be considered by the viewpoint.*

The abstract specification of a viewpoint in Definition 1 is used to establish a clear understanding on the constituents of a viewpoint. The definition's emphasis is on the modeling level aspects of the viewpoint, i.e., that a viewpoint is defined by a modeling language that is specifically designed to capture only certain aspects of the real world. The relevant aspects and the presentation of these aspects by means of created views are delineated by stakeholder concerns. Following the discussion on formalized specifications in section 4, a viewpoint specification should emphasize on a formalization of syntax, semantics and notation.

A view (Definition 2) is then defined as a result of the mapping of some real world phenomenon into a graphical visualization, i.e., a model of the relevant aspects of the real world, by means of a viewpoint instance. Hence, a view follows the guidelines of a viewpoint, i.e., semantics, syntax, and notation of the viewpoint specification are considered.

Definition 2 (View) *A view is the result of the creative process of a modeler who creates an artifact as a mapping of relevant aspects of the real world by following the guidelines of, and utilizing the modeling language specified by, a viewpoint. The view establishes a graphical representation of certain aspects of the real world by means of a viewpoint instance.*

Definitions of "multi-view modeling" are manifold (cf. section 2.2.2). Most of the definitions found in relevant literature have in common that they emphasize on the *difference* of the views and the *separation* of concerns. By contrast, the proposed generic definition of multi-view modeling considers the syntactic and semantic relationships between the multiple views. Hence, the multiple views have no means on their own, they are considered with their multifarious relationships to each other.

Definition 3 (Multi-View Modeling) *Multi-view modeling enables humans and machines to interact from multiple semantically or syntactically dependent perspectives with different views of a modeled artifact.*

Definition 3 not only centers the modeled artifact, it also emphasizes on the dependencies between the views. Views are therefore created by taking different perspectives, i.e., viewpoints, on a modeled artifact. These perspectives are related, they often impose syntactic and/or semantic overlaps. Multi-view modeling methods, and the corresponding models, should be specified in way that allows not only processing by humans but also by machines. Hence, views should

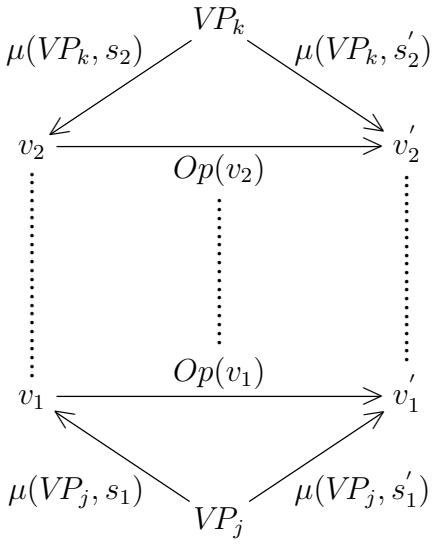


Figure 24: Formalized specification of multi-view modeling (cf. BORK AND KARAGIANNIS (2014)

be instantiated from formally specified viewpoints (cf. section 4). In the following, a more formalized specification of multi-view modeling is given.

Figure 24 illustrates the interplay of viewpoints and views using a formalized notation[22]. Compared to models, which are instances of a meta model, the constituents of a view (v) are specified by a viewpoint (VP). View v_1 is an instantiation of the Viewpoint VP_j in state s_1, View v_2 is an instance of Viewpoint VP_k in state s_2. The instantiation is generated using the μ operator that is applied to a certain viewpoint. In analogy to the FDMM formalism (FILL ET AL., 2012), the μ operator instantiates a viewpoint into a view in a certain state, e.g., defining the existing object types, data types, and attribute values[23]. Modeling operations, e.g., creation, deletion, or update of modeling objects or relations, are depicted by Op. Aplpication of modeling operations transform a view of a certain state v_i into a new state v_i'.

The aim of multi-view modeling is to provide not only consistency between multiple views - the dotted lines between v_1 and v_2 and v_1' and v_2', but also a semantic equivalence between the modeling operators Op, applied to multiple views. The former is concerned with consistency in a static manner, whereas the latter considers multi-view consistency in a dynamic manner. This formalized and generic specification will be used as an semantic anchor for the specification of multi-view modeling characteristics and different multi-view modeling principles in the following.

[22] The visualization is an analogy to the view-update problem, described in relational databases in the late 1980s KELLER (1985)

[23] The illustration uses a more compact notation for μ. It therefore omits the attributes of the state operator s.

5.2 Viewpoint Determinants

In this thesis, a lot has been discussed already on the concept of views and viewpoints. However, the origins of a viewpoint, i.e., the determinants for establishing a viewpoint where not discussed thoroughly. In order to grasp an idea on the origins of viewpoints and possible viewpoint classifications, a look into the related literature is insightful. The aim of this classification is to enable method engineers in identifying and managing viewpoints, and model users in selecting appropriate viewpoints.

5.2.1 Related Work on the Origins of Viewpoints

Teeuw introduced three dimensions for a high-level structuring of conceptual models: *Abstraction level*, *Level of detail*, and *Specifications* (see Figure 25). In this regard, different viewpoints may origin from different levels of abstraction, e.g., a conceptual viewpoint or a technical viewpoint; different levels of detail, e.g., a comprehensive viewpoint or a detailed viewpoint; or from different specification foci, e.g., a behavior viewpoint or a structural viewpoint.

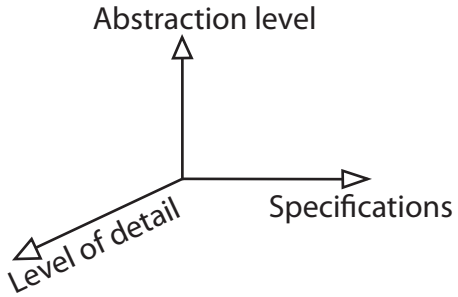


Figure 25: High-level structuring of a conceptual model (TEEUW AND VAN DEN BERG, 1997)

In the domain of software architecture, a view is derived from a set of concerns it addresses, hence, indirectly from a stakeholder who defines these concerns. LANKHORST (2009) classified a viewpoint along two dimensions: *purpose* and *content*. The former is further decomposed into the categories *designing*, *deciding*, and *informing*, whereas the latter is further decomposed into *details*, *coherence* and *overview*. Bergmann and Wilke argue, that *"different levels of abstraction require different representation languages, one for each level"* (BERGMANN AND WILKE, 1996, p. 29).

Atkinson's Orthographic Software Modeling approach has an emphasis on the specification of viewpoints. Viewpoints are structured along predefined but extendable dimensions. Each dimension is decomposed into a set of *divisions*, some of them hierarchically structured, others not. (ATKINSON AND STOLL, 2008, p. 95) introduced the following set of viewpoint dimensions:

Composition This dimension enables the hierarchical structuring of viewpoints by means of composition relationships. Different hierarchical divisions of this dimension may be e.g., the enterprise viewpoint, the business process viewpoint of one department of the enterprise, and the workplace model viewpoint for a certain employee in a certain department.

Abstraction This dimension covers the degree of platform-independence of a viewpoint. Due to the background of the OSM approach, coming from the software engineering domain, the abstraction dimension is initially introduced by defining three divisions of the dimension: specification, realization, and implementation.

Projection The *"types of information contained in a view"* (ATKINSON AND STOLL, 2008, p. 95) is considered in the projection dimension. Initially, *structural*, *functional*, and *behavioral* projection viewpoints have been identified.

Grundy et al. specified three classes of viewpoints in multi-view specification of object-oriented systems (GRUNDY ET AL., 1991; GRUNDY AND HOSKING, 1993): (1) *Base views*, (2) *Subset views*, and (3) *Display views*. The authors argue, that there is always one canonical model, combining all information in the multiple viewpoints. This canonical model is referred to as the *base view*. Each viewpoint, depicting only a certain subset of the *base view* is considered as *subset view*. The third category, *display views*, covers the viewpoint-dependent visualization of aspects in *subset views*. The user solely interacts with these *display views*. All performed modifications are transformed into changes on the *subset view* and the *base view*, respectively.

Along with the ideas proposed by Grundy, the research of MUNKER ET AL. (2014) should be outlined. They also proposed hierarchical structuring of viewpoints, but by means of thematically structuring the constituents of a view, by contrast to structuring viewpoints on a meta level. The authors propose to use a thematic structuring for multiple *thematic viewpoints* in SysML (OBJECT MANAGEMENT GROUP (OMG), 2012b). All thematic viewpoints are integrated through one *holistic system model*. A procedure for generating viewpoints is also introduced. It is composed of three steps (cf. (MUNKER ET AL., 2014, p. 534f.)): (1) model the component structure using SysML composition relationships, (2) (optionally) apply view-specific stereotypes on components, and (3) model view-specific thematic structures using SysML aggregation relationships.

Reineke et al. introduced a more formal way of describing views (i.e., viewpoints). Generally, the authors define a system as a *set of behaviors*. Using this definition, three categories of views can be defined as follows (REINEKE AND TRIPAKIS, 2014, p. 3): *1. Subset View:* Subset views contain only a subset of all behaviors of the system, the view is created of. All parts of the included behaviors are complete. *2. Projection View:* A projection view further constrains the subset views by including only a subset of the parts of a behavior of the system. *3.*

Transformation View: Creating a view using a certain transformation algorithm, not necessarily a projection, characterizes the third category.

5.2.2 A Multi-Dimensional Classification Schema for Viewpoints

The understanding of viewpoint determinants followed in this thesis is a broader and informal one. The aim is to enable a viewpoint to be generally composed of everything that is necessary to an addressee of that viewpoint, i.e., a set of stakeholders. Therefore, the constituents of a viewpoint are determined by the needs and concerns of the stakeholders addressed by a viewpoint.

Hence, on a general basis, viewpoints can be distinguished by: i) the concerns they frame, ii) the abstraction level they apply, iii) the stakeholders they serve, iv) the notation they utilize, and v) the level of detail they visualize. To grasp a general understanding of this admittedly very open definition of a viewpoint, Figure 26 might be helpful. Similarly to the concept of multi-dimensional data structures, utilized e.g., in data warehouse systems, that structure data along multi-dimensional hierarchies, a multi-view model is composed of viewpoints, originating from a combination of this multi-criteria determinants. Figure 26 uses the determinants to establish a multi-dimensional structure of the viewpoints. Notably, not every dimension has the same number of sub-dimensions. Moreover, not all cells in this conceptual "cube" need to be represented by a viewpoint in a multi-view modeling method.

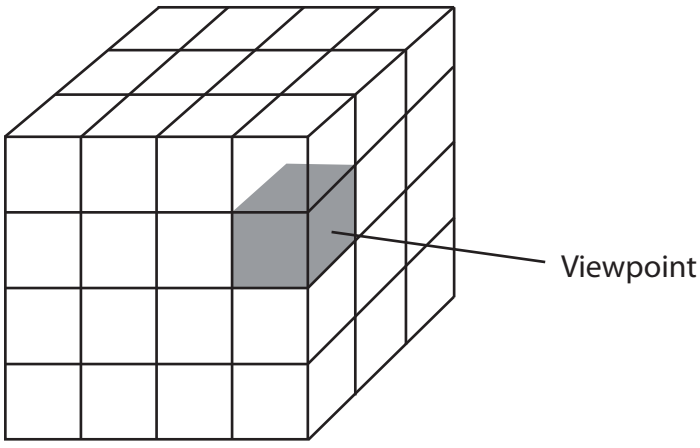


Figure 26: Multi-dimensional classification of viewpoint determinants

As the cube also indicates, a viewpoint, respectively the view created accordingly, has relationships and dependencies to the other viewpoints and views. The specification of views and the relationships between views will be investigated in the following. When it comes to the design of a multi-view modeling method, i.e., the specification of the viewpoints, some guidelines should be followed. In 2014, ATKINSON AND TUNJIC (2014) introduced a set

of seven "informal goals and properties", aiming at optimizing the orthogonal viewpoints. Although specific for the OSM approach, some of the proposed aspects can be generally applied to multi-view modeling methods: 1) it should be possible to completely represent the subject using views; 2) views should have an optimal size; 3) the number of viewpoints should be minimal; 4) static dimensions should have the similar numbers of dimension values; 5) the number of dimension values should be optimal; 6) views should have minimal overlap; and 7) there should be a minimal number of viewpoint-free cells.

Independently from the determinants of the viewpoint, an emphasis should be on its formalized specification. In the following, the relationships between meta models, capable for formally specifying the modeling language of a viewpoint, and viewpoints, are investigated, hereby identifying a dichotomy of meta model based viewpoint integration.

5.3 Viewpoint Architecture Framework

Having discussed the origin and the determinants of the multiple viewpoints on a semantical and mereological level, this section embraces on the syntactic and formalized specification of viewpoints. Hence, the relationships between viewpoints and meta models will be discussed in the following. In section 2.2.1 the similarities of multi-view modeling to relational databases has been discussed. It is therefore no surprise, that there are also similarities in the specification of the syntactic viewpoint relationships. *"Integrating heterogeneous metamodels and instances bears similarities to the well-known problem of schema integration of heterogeneous databases ((*REDDY ET AL., *1994;* SHETH AND LARSON, *1990))"* (BURGER, 2013, p. 1:4).

In order to specify different relationship classes, a generic architecture framework is first introduced in the following. SINZ (1997, 2002) introduced a generic architecture framework for information systems, feasible for establishing a generic and comprehensive classification scheme referred to later on. The goal of this framework is to enable the analysis and specification of complex information systems architectures. Figure 27 illustrates the components of the generic architecture framework. The framework utilizes a layered approach for handling the complexity of information systems that is introduced in the following:

- Definition of *model levels*, structuring the model system (cf. section 2.1.1). Every model level describes the information system comprehensively, thereby taking a certain perspective on the information system. The different perspectives are aligned to the different goals pursued during the modeling process. Common perspectives, are the differentiation according to task and resource, or according to an inside and outside perspective (cf. section 7.1 and Figure 41 for the application of these perspectives in the SOM method). The construction rules for each model level are defined by a corresponding meta model.

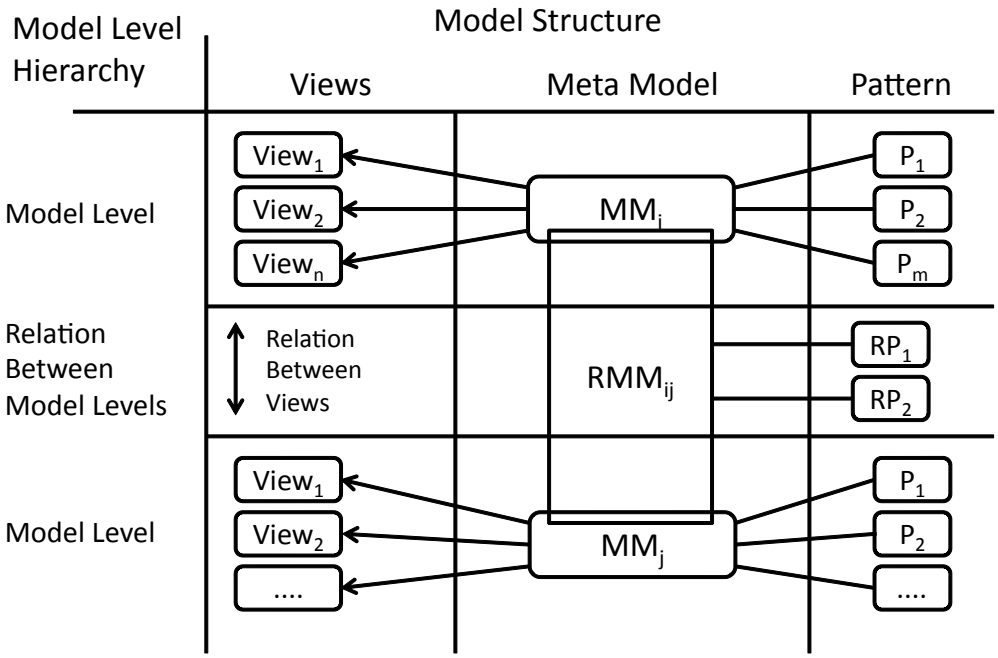


Figure 27: Generic architecture framework (SINZ, 2002, p. 876)

- Definition of *views* for each model level. Views are specified by defining a projection operator based on the *meta model* of the corresponding model level. Each view describes usually a partition of the model. All views are integrated through the common meta model. The combination of the views on one model level gives the comprehensive model on that level. Common views, according to Sinz are e.g., data view, function view, structure view, or behavior view (cf. section 7.1 for the application of the structural and behavioral view in SOM business process modeling).

- Definition of *relationships* between model levels. For each pairwise relation between model levels, relations are defined by specifying a relationship meta model (RMM). A RMM_{ij} relates meta model elements of meta model i (MM_i) to meta model elements of meta model j (MM_j). The specification of these relationships between the meta models of different model levels integrates the comprehensive model system. Furthermore, assignments and transformations between model levels can be defined (SINZ, 1997, p. 6).

- Definition of *patterns* (P) and *relationship patterns* (RP) for each model level and between model levels, respectively. Patterns are used to limit the valid structures of a model level, e.g., based on heuristic modeling experience or as a means of structural integrity constraints (SINZ, 1997, p. 4).

The existence of the relationship meta models (RMM) does not only contribute to an integration of the meta model, but also to the definition of relationships between the views. These

relationships are transitively defined by the relationship between the corresponding meta models (SINZ, 2002). However, Sinz already stated that *"In many modeling languages only more or less isolated sub-meta models for the individual views are given. By contrast, an integrated meta model is not available. From a methodological point of view, this represents a serious deficit, since the alignment of the individual views is prevented or at least hindered"*[24] (SINZ, 2002, p. 877).

The positive effects of an integrated meta model are manifold. The most important ones are described briefly: The definition of an Integrated Meta Model (IMM) defines relationships not only between the meta models but also on the views, derived from the meta models. Consequently, syntactic dependencies can be defined once on meta level and are effective for any created instance. Adding a new meta model to the model system, and therefore to the IMM, enables immediate transformation and alignment between all other meta model instances and the new one. Moreover, the definition of an IMM facilitates a general understanding and documentation of the complex multi-view architecture. Zachman already stated for information systems architectures in 1987, *"since the technology permits "distributing" large amounts of computing facilities in small packages to remote locations, some kind of structure (or architecture) is imperative because decentralization without structure is chaos"* (ZACHMAN, 1987, p. 454). The same holds for the decomposition of the model system into model levels and views. As the integration of meta models plays an important role, the different approaches for meta model integration in the context of **integrated multi-view modeling** are now discussed.

5.3.1 A Dichotomy of Integrated Multi-View Modeling

In case of multi-view modeling, the tight coupling of the information systems architecture presented before is not feasible. By contrast, more flexible approaches to integrate multiple views, i.e., models, need to be considered. In the following, a dichotomy of integrated multi-view modeling is established.

5.3.1.1 Meta Model Integration

Figure 28 illustrates the case of an already existing integrated meta model. This case refers to the generic architecture framework for information systems introduced in section 5.3. The *object system*, i.e., the relevant part of the real world, is mapped into a *model system* (cf. section 2.1.1) during the modeling process. Integrity of the *model system*, and the multiple views derived, is established by the *integrated meta model*. Views on the model system are defined as projections, that can be applied to the integrated meta model. The integrity of the views is therefore transitively realized by the the design of the meta models.

[24] English translation by the author.

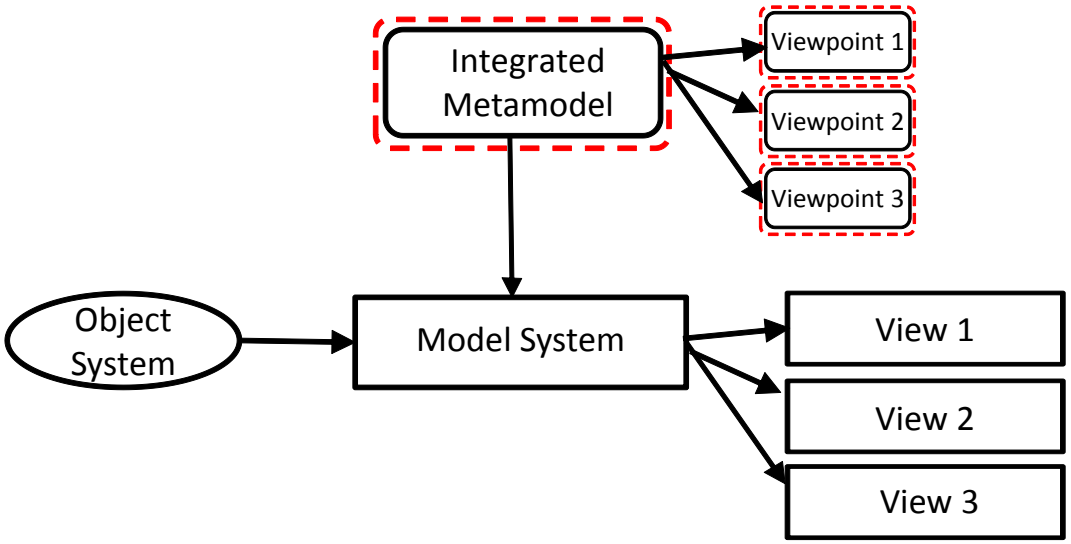


Figure 28: Meta model integrated multi-view modeling (cf. BORK AND SINZ (2011b))

5.3.1.2 Viewpoint Integration

As already stated by SINZ (2002), the common case is that, where method engineers are forced to integrate "more or less isolated sub-meta models". Hence, each viewpoint has its own meta model that needs to be integrated towards an integrated multi-view modeling method. Figure 29 illustrates this case. The object system is mapped into several views whereas each view is constructed according to a viewpoint. The viewpoint specification itself is based on a specific meta model. The model system is therefore not integrated by the meta model. The integration of the model system, by means of the multiple views visualizing it, has to be realized by a previous integration of the corresponding viewpoints.

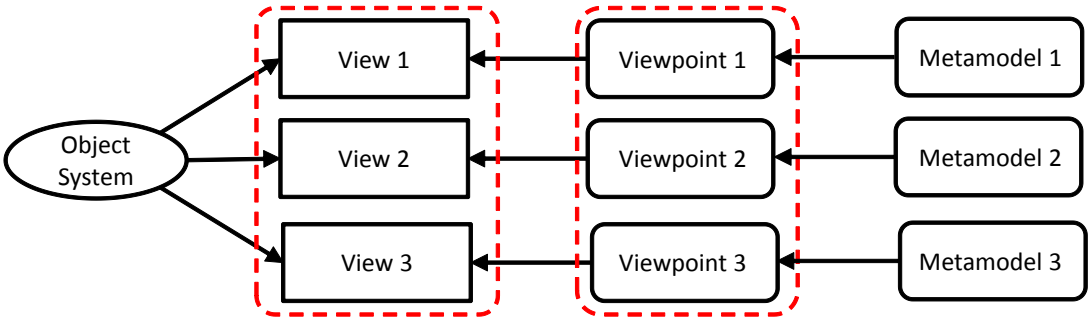


Figure 29: Viewpoint integrated multi-view modeling (cf. BORK AND SINZ (2011b))

5.3.2 Discussion

There are different ways of integrating multiple views. The presented dichotomy compared meta model integrated and viewpoint integrated approaches. The latter enables more flexibility, however, at the cost of specification effort. Whenever a new viewpoint is introduced it has to be analyzed and probably integrated with all other viewpoints, whereas in the former, it only has to be integrated with the integrated meta model. Both approaches can be aligned to the categories identified by the IEEE: *"There are two common approaches to the construction of views: the synthetic approach and the projective approach. In the synthetic approach, an architect constructs views of the system-of-interest and integrates these views within an architecture description using model correspondences. In the projective approach, an architect derives each view through some routine, possibly mechanical, procedure of extraction from an underlying repository"* (IEEE, 2011, p. 22).

Boulanger et al. contributed to this understanding by stating: *"The first and most common one is to consider that views are projections of a hybrid reference model. In this approach, the reference model aggregates all the information about the system. Views are queries on the reference model; they perform projections hiding irrelevant information when studying a particular aspect of the system. The second approach uses views as partial definitions or expected observations of the system"* (BOULANGER ET AL., 2010, p. 313).

Albeit all the benefits of the projective approach (i.e., the meta model integrated approach), it must be emphasized, that the relevant literature not considers the actual creation and processing of the models. The authors assume, that with this multi-viewing approach all integrity checks and inconsistencies are solved by design. However, while developing multi-view modeling tools this assumption does not hold. In the following, two approaches are introduced that emphasize on the interactions between modelers and multi-view modeling tools - the way of carrying out multi-view modeling.

5.4 Application Scenarios for Multi-View Modeling

The preceding sections concentrated on static aspects of multi-view modeling, e.g., the determinants of viewpoints or the integration of viewpoints. What is missing is a precise specification of the behavioral aspects of multi-view modeling. Hence, in the following the focus is on applying multi-view modeling. As already illustrated in the generic multi-view modeling framework in Figure 24, the operations applied in a multi-view setting play a mature role. Figure 30 provides a conceptual model for the specification of multi-view modeling operators.

Figure 30 is motivated by the definition of *modeling transactions* (cf. Definition 4 (BORK AND SINZ, 2010, p. 6)). The conceptual model focuses on the multi-view characteristic by relating modeling operations to an execution context. Hereby, an emphasis is on the specification

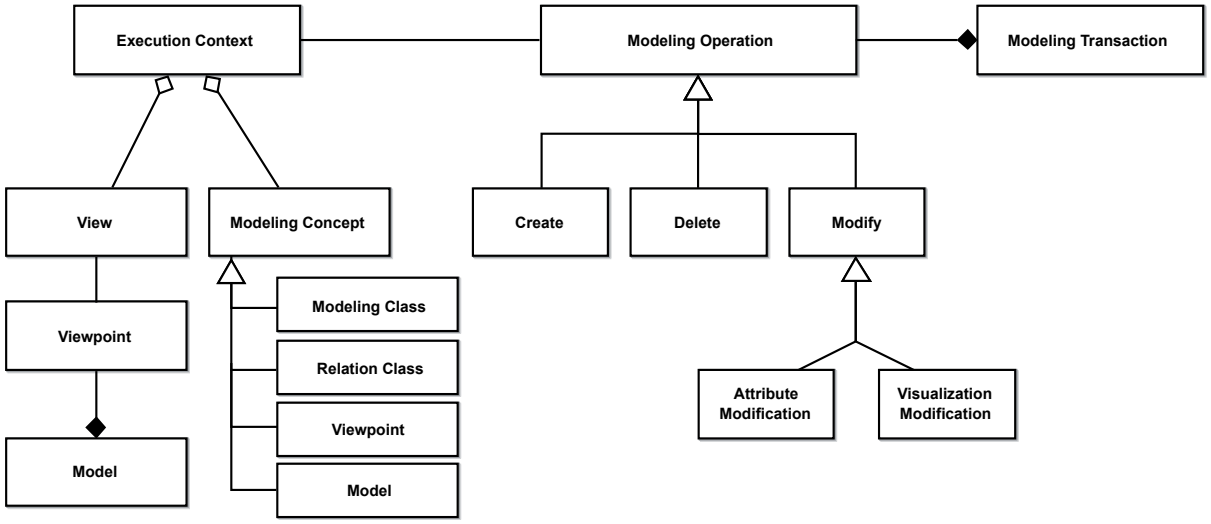


Figure 30: Multi-view modeling operators

of the viewpoints and modeling concepts that are considered by the execution of an operation.

A *modeling transaction* is composed of simple *modeling operations* like *create*, *edit* or *modify*, whereas the latter one is further decomposed in the context of graphical modeling by separating *attribute modifications* from *visualization modifications* (e.g., moving objects on the drawing area). Every modeling operation is executed in an *execution context*, relating the modeling concept the modeling operation executed at (i.e., a modeling class, a relation class, a viewpoint, or the whole model) to the view, viewpoint, or model.

Definition 4 (Modeling Transaction) *A modeling transaction is composed of a sequence of editing operations which transform a consistent state of the model into a new state, which again is consistent according to syntax and semantics.*

The aim of multi-view modeling methods and multi-view modeling tools should be on the specification of modeling transactions as defined in Definition 4 using the conceptual model visualized in Figure 30. The relationships between the views are manifold. Hence, the application of modeling operators is viewpoint-dependent and might require the application of *syntactically completely different* but *semantically equivalent* modeling operators to other viewpoints in order to maintain consistency. In the following, two classes of consistency management approaches are introduced.

5.4.1 Multi-View Modeling by Design

This category considers all modeling methods that emphasize on multi-viewing aspects from the very beginning, i.e., by the design of the method. The multiple viewpoints, the dependencies between the viewpoints, and the procedure of generating and manipulating multi-view models are considered in the method specification, generally in a formalized manner. This formalization eases the transformation of the specification into supporting modeling tools.

Integration of the viewpoints can be achieved by providing one comprehensive base meta model (i.e., an integrated meta model) that integrates the multiple viewpoint meta models (cf. meta model interfacing EMERSON AND SZTIPANOVITS (2006)). The creation of models according to the *by design* approach utilizes simultaneous modeling of multiple views that are integrated and kept consistent.

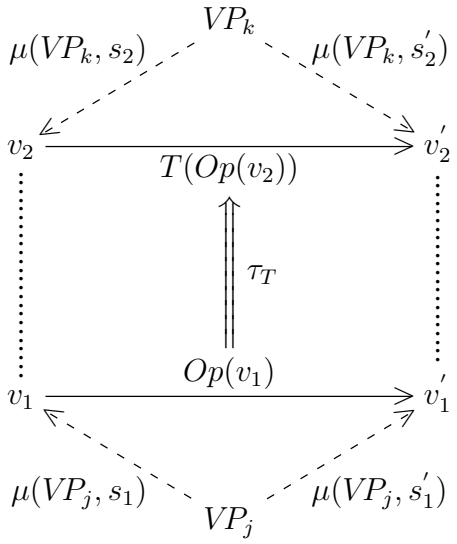


Figure 31: Multi-view modeling by design

Figure 31 illustrates the **multi-view modeling by design principle** by adopting the generic conceptual model visualized in Figure 24. From each viewpoint, a view is instantiated by the μ operator. Following the *by design* approach, both created views, i.e., v_1 and v_2 are consistent. The application of a modeling operator (Op) on v_1 transforms view v_1 into a new model state, referred to as $v_1^{'}$. A modeling tool supporting multi-view modeling by design must provide a transformation of operator Op, $(T(Op))$, that can be applied on view v_2 in order to transform v_2 into a new state $v_2^{'}$, which is again consistent to $v_1^{'}$. $(T(Op))$ has to be specified on a viewpoint-to-viewpoint basis. A modeling tool should provide mechanisms to translate modeling operations (more generally changes initiated by the modeler) on one view to corresponding modeling operations on other views. In the following, this translation is referred to as *transition translation* (τ_T).

Definition 5 (Transition Translation (τ_T)) *Transition Translations (τ_T) transform the modeling operations performed by the modeler on one view into semantically equivalent modeling operations that need to be applied to other views in order to maintain a consistent multi-view model.*

Considering the conceptual model for multi-view modeling operators (cf. Figure 30) the translation needs to be considered in its execution context. Depending on the source (e.g., the viewpoint that has been changed by the modeler), the modeling operation has been executed at, the target (e.g., a different viewpoint) and the corresponding semantically equivalent modeling operation must be identified. Hence, there is generally no one-to-one mapping of modeling operations to viewpoints.

Realization of Transition Translations

For the conceptual and technical realization of the transition translations in modeling tools, several technologies and existing techniques can be applied. FINKELSTEIN ET AL. (1992) propose the usage of *work records*. These work records track the actions performed by the modeler while following the *work plan*. It comprises *assemble actions*, *check actions*, *viewpoint actions*, and *guide actions* (FINKELSTEIN ET AL., 1992, p. 12). Such work records are on an informal level, however they foster the comprehensive conceptual specification of transition translations if they would be combined with the corresponding (τ_T) transformations.

Technically, (τ_T) can be realized by referring to constraint languages like the Object Constraint Language (OCL), the IBM Model Transformation Framework[25], or the incremental model transformation framework (VARRÓ AND BALOGH, 2007) (for an overview of model transformation approaches by means of a taxonomy, see (CZARNECKI AND HELSEN, 2003)). Moreover, if all modeling operators could be specified as graph rewriting rules, Triple-Graph Grammars can be utilized as well (GIESE AND WAGNER, 2009; LAUDER ET AL., 2012).

5.4.2 Multi-View Modeling by Generation

The *multi-view modeling by generation* principle summarizes all approaches where multiple viewpoints are depending on each other by a temporal relationship. Consequently, a certain viewpoint can only be created using a generative transformation of an already modeled viewpoint, or several viewpoints. By contrast to the *by design* principle, multiple views are processed

[25] Model Transformation Framework, `http://www.ibm.com/developerworks/rational/library/05/503_sebas/` DEMATHIEU ET AL. (2005), last checked: 2015-04-10

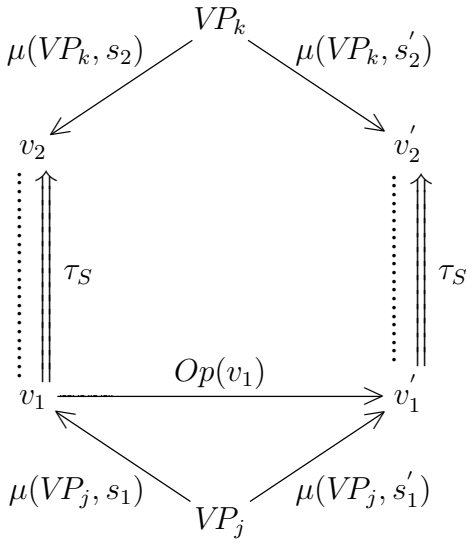


Figure 32: Multi-view modeling by generation

in sequence. Modeling operations performed on a certain viewpoint are not propagated to other viewpoints. Instead, the view is transformed as a whole by translating the current state of a viewpoint into a semantically equivalent state of a different viewpoint, hereby generating the other view. This transformation is referred to as *state translation* (τ_S).

Definition 6 (State Translation (τ_S)) *State Translations (τ_S) transform the state of an entire view into a semantically equivalent state of a different view in order to maintain a consistent multi-view model.*

Figure 32 illustrates the multi-view modeling by generation principle in more detail. It can be derived, that modeling operations, executed on a certain view do not affect other views. $Op(v_1)$ is not automatically translated into a semantically equivalent operation that needs to be applied to view v_2. Consistency between views must be ensured using state translations. Every change performed by the modeler in one view inevitably results in an inconsistent multi-view model. The modeler (or the modeling tool) is required to execute the state translation another time in order to establish consistency.

Most code-generation approaches from the computer science domain that utilize model-driven development of software systems can be categorized as multi-view modeling by generation approaches. These approaches do not allow round-trip engineering. Changes in the technological viewpoint (e.g., source code) are not reflected in the conceptual viewpoint (e.g., class diagram or software architecture).

Realization of State Translations

The conception and technical realization of state translations (τ_S) is more intuitive compared to the transition translation. Most of the model-driven development and model-driven engineering approaches utilize conceptual and/or technical specifications for the dependencies between multiple model (i.e., views), e.g., MOFLON (AMELUNXEN ET AL., 2008). In (BRAATZ AND BRANDT, 2011, 2014) the utilization of the Resource Description Framework (RDF) for rule-based integration of multiple domain-specific modeling languages is elaborated. Hence, meta model based model transformations can be utilized, hereby relating the elements of multiple meta models to each other.

Query View Transformation (QVT) (OBJECT MANAGEMENT GROUP (OMG), 2015) specifications can be utilized for the purpose of state translations as exemplified by ROMEIKAT ET AL. (2008). Moreover, Triple Graph Grammars can also be utilized for the realization of state translations.

Table 13 summarizes and compares the characteristics of by design and by generation multi-view modeling in a comprehensive way.

Table 13: By design and by generation multi-view modeling principles

	by design	by generation
Viewpoint specification	Formalized specification of all viewpoints and the relationships between the viewpoints e.g., using meta models	Viewpoints are independently specified. All relationships between viewpoints must be added manually
Integration	Meta Model Integration	Viewpoint Integration
View creation	Simultaneous initialization	Generative, ad-hoc initialization
Translation τ	Transition translation (τ_T)	State translation (τ_S)
Model editing	Simultaneous editing of multiple views	Editing of single views

5.5 Summary

The preceding section introduced a sound theoretical and conceptual foundation for conceptual modeling using integrated multiple views. These foundation establishes the understanding that

is used to follow the motivation on the MUVIEMOT method that will be introduced in the following section. The MUVIEMOT method is specifically designed to capture the identified characteristics and to codify them by means of conceptual models. The MUVIEMOT method aims at the model-driven specification of conceptual designs for multi-view modeling tools.

6 Conceptual Design of Multi-View Modeling Tools: The MuVieMoT Method

> *Designing is not the act of drawing a diagram: a diagram simply captures a design. If you follow the work of any engineer, software, civil, mechanical, chemical, architectural, or whatever - you will soon realize that the one and only place that a design is conceived is in the mind of the designer.* Booch (1991)
>
> *(Grady Booch)*

Now that the benefits of a formalized method specification and the foundations of multi-view modeling methods have been established, the question that remains to be unanswered is how to capture these aspects in a conceptual modeling method, enabling method engineers and tool developers in conceptually designing multi-view modeling tools. In the following, the MuVieMoT modeling method is introduced. The method is aimed at bridging the gap between a multi-view modeling method on the one hand, and the conceptual design of a corresponding multi-view modeling tool on the other. The bridging is realized by a procedural approach, guiding the conceptualization process in a step-wise and model-driven manner.

This section is organized as follows: Section 6.1 motivates the development of MuVieMoT by pointing to the lack in support for method and tool engineers in designing multi-view modeling tools. The requirements for such a method are gathered in section 6.2. The components of MuVieMoT, aimed at covering these requirements, are then introduced in section 6.3. In section 6.4, the MuVieMoT method is contrasted with the requirements, enabling a first evaluation of the method. Moreover, a SWOT analysis is performed and the constituents of the MuVieMoT method are contrasted to the research questions this thesis is grounded on.

6.1 Motivation

Referring to section 2.2.1, multi-view modeling approaches have been utilized in divers domains and incarnations. However, most of the approaches consider multi-viewing on an ontological or methodological level in the context of enterprise architecture frameworks or software engineering, taking multi-viewing as a means of facilitating comprehension of complex systems and separation of concerns. By contrast, the MuVieMoT method approaches multi-viewing from a meta modeling perspective, targeting at method engineers and tool developers. Hence, multi-viewing is investigated not only from a conceptual but comprehended with a practical perspective. Moreover, viewpoints are considered first-class citizens during the conceptual design

of multi-view modeling tools.

In the past, most of the multi-viewing approaches have been realized solely on a theoretical, at most on a conceptual level. Only few approaches have realized a prototypical implementation. However, the availability of appropriate modeling tools considerably fosters acceptance, propagation, and finally application of modeling methods. This all the more holds for multi-view modeling methods. The number of modeling views that need to be kept consistent and simultaneously processed by the modeler exceeds the cognitive capabilities of human beings. Moreover, modeling method specifications, independently from their degree of formalization, lack at codifying the specifics of multi-viewing. Modeling operations, modeling procedure and algorithms & mechanisms need to be considered differently, if a multi-view modeling tool is to be designed. The design of comprehensive, intuitive usable and consistency preserving multi-view modeling tools is very challenging (GRUNDY ET AL., 2013) and not considered in research appropriately at the time this thesis was written.

The aim of the MUVIEMOT method is to lower the level of abstraction for modeling tool design. The method builds on the foundations of multi-view modeling methods and discusses them in a meta modeling context. Method engineers and tool developers may consider the method during the process of conceptually designing and implementing modeling tools. In this regard, MUVIEMOT can be considered a domain-specific modeling method for the conceptual design of multi-view modeling tools.

Why a Model-Driven Approach?

Any novel methodical approach needs to be evaluated considering its efficiency, suitability and applicability. It is common sense that models help to capture the relevant aspects of the reality by providing a more suitable (e.g., higher) abstraction level. *"The use of such a higher level graphical language has the distinct advantage that it removes complexity from the design process and allows a more intuitive method of using graphical representations for the design of the system"* (WOOD ET AL., 2008, p. 1357).

The method engineering approach tries to realize the benefits of modeling methods and model-driven development approaches (cf. section 2.4). The different steps of the approach are supported by specifically designed conceptual modeling languages (cf. section 2.3). Hence, the languages are aimed to provide the most suitable abstraction level and utilize modeling concepts that are strictly aligned to, or derived from the application domain. In the context of MUVIEMOT, this domain comprises concepts of multi-view modeling methods, meta modeling and requirements engineering. Moreover, as discussed in more detail in section 8, the created models not only enable the specification and interpretation of a conceptual tool design, they also enable the model-driven development of corresponding modeling tools.

It might be somehow confusing, but the arguments for utilizing multi-viewing also hold

for the MUVIEMOT method. MUVIEMOT itself can be considered a multi-view modeling method. Each step of the method considers a specific viewpoint on the multi-view modeling tool; thereby utilizing a specifically designed modeling language, targeting at the specific needs of the intended audience of the viewpoint (cf. 5.2).

Adopting the generic benefits of utilizing a model-driven development approach introduced in (GUPTA AND KUMAR, 2014, p. 301), the most relevant benefits in the context of the MUVIEMOT method are as follows:

- MUVIEMOT models are used as primary artifacts of modeling tool construction.
- MUVIEMOT models are used to provide better understanding of the domain that helps tool developers to realize correct and consistency-preserving multi-view modeling tool implementations more efficiently.
- MUVIEMOT models facilitate early stage evolution of a software system that helps to reduce the number of errors in the final modeling tool implementation.
- MUVIEMOT models bridge the gap between method owner and tool developer.
- MUVIEMOT modeling procedure enables to handle the complexity of modeling tool development by guiding the specification process along a predefined sequence of steps.

6.2 Requirements

In the following, requirements for a modeling method, capturing the specifics of multi-view modeling methods and the practical aspects of tool development are identified. The requirements are built on several pillars: i) requirements found in a comprehensive literature review on multi-view modeling methods and corresponding tools; ii) requirements originating from the experience of developing multi-view modeling tools; iii) requirements from the experience of using several multi-view modeling tools, targeting at usability, and iv) requirements for modeling languages in general.

It must be stated, that general requirements for the development of modeling methods are not considered in depth now. The focus is on the specific requirements for a modeling method targeting at the specification of a conceptual design for multi-view modeling tools. Generic modeling method requirements are discussed thoroughly by e.g., FRANK (2011c,d) or the research of BRINKKEMPER (1996) on method engineering.

6.2.1 Requirements from Relevant Literature

The missing formalization on the specification of the RM-ODP motivated Akehurst already 2004 to point to the research gap between a multiple-viewpoint method (RM-ODP) on the one

hand and the specification of an accordingly designed modeling tools on the other (AKEHURST, 2004). The author argues for at least three inevitable constituents of a multi-view method specification: (1) Precise definitions of the *viewpoint language concepts*, (2) precise specification of the *correspondences between concepts* in each viewpoint, and (3) precise specification of *example notations* for each viewpoint.

Akehurst and Bobrik et al. moreover emphasized on the importance of visualization in multi-view environments (AKEHURST, 2004; BOBRIK ET AL., 2007). The specification of viewpoint-dependent visualizations should therefore be supported. This would allow the alignment of the visualization of common aspects to the characteristics (e.g., audience, abstraction level, content, intention) of specific viewpoints. GRUNDY ET AL. (2013) (based on initial work by (SUTCLIFFE, 2002)) identify key requirements for domain-specific visual language specifications.

LIT-1 A specification should consider the modeling language of the viewpoints by means of syntax, semantics, and notation.

LIT-2 A specification should comprise the relationships between modeling language concepts of multiple viewpoints.

LIT-3 A specification should emphasize on the notation of the modeling language concepts. More precisely, it should be possible to specify a viewpoint-dependent visualization of a concept.

LIT-4 The modeling languages should be comprised of concepts in an appropriate abstraction level, originating from the domains of multi-view modeling and conceptual tool design.

LIT-5 Appropriate and intuitive understandable visualizations should be provided for each element of the modeling languages.

LIT-6 Whenever appropriate, multiple visualizations, e.g., graphical model visualizations complemented by tabular visualizations, should be supported in order to increase expressiveness and readability with respect to different model users and model processing scenarios (cf. (VESSEY, 1991)).

LIT-7 The modeling languages should be specified in a formalized manner. Whenever possible, at least the syntax, at best also the semantics of the modeling method should be specified in an unambiguous way (cf. section 4).

According to BORK AND SINZ (2013) the way multi-view modeling is actually applied is not considered in method specifications, yet. In section 5.4 *multi-view by design* has been distinguished from *multi-view by-generation*. In the former case, all views are kept consistent by *transition translations* (cf. Definition 5), transforming the modeling operations performed

by the modeler on one view into semantically equivalent modeling operations that need to be applied to other views. By contrast, the latter case allows temporary inconsistencies between views. Views are only consistent at the time the modeler triggers a transformation, referred to as *state translation* (cf. Definition 6). Grundy et al. emphasize also on the behavioral aspects including *"dynamic and interactive tool effects such as event and constraint handling for both model and view manipulations and automated operations or processes"* (GRUNDY ET AL., 2013, p. 489).

LIT-8 Different ways of carrying out multi-view modeling should be supported. Therefore, establishing the way modelers should interact with the multiple views and how interactions with one view are semantically transformed and applied to other views.

LIT-9 Accordingly to the former requirement, synchronization mechanisms need to be specified in order to keep the views consistent.

Several authors have emphasized on supporting method engineers in defining viewpoints (GRUNDY ET AL., 2008, 2013) and an optional navigation between viewpoints, realized by specifying hierarchical relationships between the viewpoints (ATKINSON ET AL., 2013a,b) (cf. section 3.3). This navigation hierarchy should foster understanding of the relationships between the views.

LIT-10 It should be possible to define hierarchical relationships between viewpoints, hence enabling navigation functionality.

Following (BUNGE, 1977), WEBER (1997) emphasized on *ontological completeness and ontological clarity* as major requirements for modeling methods by contrasting it to the formalized definitions of those terms adopted by requirements engineers in the software engineering domain (cf. FRANK (2011c)). Based on the mapping between an ontological model of the application domain and the syntax of the modeling language, the following anomalies might occur (cf. (WAND AND WEBER, 2002, p. 365))[26]: *Construct deficit*, an ontological concept is not mapped to a language construct; *Construct overload*, one language construct represents multiple ontological concepts; *Construct redundancy*, multiple language constructs represent one ontological concept; and *Construct excess*, a language construct has no counterpart in the ontology. If a construct deficit exists, the modeling language is considered to violate ontological completeness; if one (or more) of the other anomalies occur, the modeling language is considered to violate ontological clarity.

LIT-11 The modeling language should emphasize on ontological completeness and ontological clarity.

[26] WAND AND WEBER (2002) refer to the modeling language concepts as *construct* and to the concepts of the ontology as *ontological concept*

6.2.2 Requirements from Implementation Experience

The experience gained during the development of several modeling tools based on the ADOxx meta modeling platform serves also as a basis for practical tool development requirements. Although not scientifically valid, the experience contributes a practical perspective to the set of requirements.

EXP-1 Specification of temporal dependencies between views, i.e., viewpoint B is transformed automatically from an already modeled viewpoint A. The transformation specification should be in a formalized manner, as experience showed that often only a semi-formal or ontological specification of the transformation is given, leaving space for interpretations to the developer.

EXP-2 Specification of a mapping between modeling operations and viewpoints, they can be executed in. Experience showed that even if the modeling methods have specified the modeling operations thoroughly, the tool developer still has problems to constrain the operations to the viewpoints.

EXP-3 The effects of executing modeling operations in one view on all other viewpoints should be defined accurately. Experience showed that it is likely that modeling operators are only applicable in certain viewpoints, whereas their execution may have (conditional) effects on other viewpoints.

EXP-4 Mechanisms and algorithms should be considered in the context of multi-viewing. Specifically, view transformation, analysis, validity checks, and consistency checks should be specified.

6.2.3 Requirements aiming at Usability

In order to be applicable in an intuitive and efficient manner, usability requirements also need to be considered.

USE-1 In order to cope with the complexity and the huge amount of information that needs to be considered, decomposing the method into a procedure of several steps should be supported.

USE-2 The method should define clear backtracks enabling the modeler to move to an earlier step in the modeling procedure, perform changes in this earlier step and proceed from this updated models.

USE-3 The modeling languages should be intuitively usable. This requirements covers all facets of usability of visual modeling languages (e.g. decomposition functionality in

order to prohibit models exceeding the cognitive capabilities of the modeler (VESSEY, 1991), and Moody's principles for designing visual notations (MOODY, 2009)).

6.2.4 Modeling Language Requirements

General requirements for modeling methods and modeling languages are summarized here. Although some of them seem obvious, they contribute to a comprehensive requirements specification.

LINDLAND ET AL. (1994) introduced a systematic framework for evaluating the quality of conceptual modeling languages. The authors propose three quality criteria: *syntactic quality*, *semantic quality*, and *pragmatic quality*. In (TEEUW AND VAN DEN BERG, 1997), the semantics and pragmatics criteria of this framework have been further decomposed into the following six quality criteria (based on work of BLAAUW AND BROOKS JR. (1997) and SINDEREN (1995)): *Completeness, Inherence, Clarity, Consistency, Orthogonality, Generality*.

LANG-1 If the relevant aspects of the domain are still too complex, and the resulting overarching models would overwhelm the modeler, decomposition of the method (and the model) into several steps (and models) should be provided. Each step is supported by a modeling language that only captures a sub-set of the overarching model.

LANG-2 The steps followed during the application of the method should be precisely defined, i.e., the tasks involved in a step, the objectives of the step, and the dependencies to the preceding and succeeding steps.

LANG-3 The different steps should be semantically integrated. Hence, reuse of already defined aspects should be supported.

LANG-4 *Completeness:* The modeling language should provide concepts to capture all relevant aspects of the real world.

LANG-5 *Inherence:* The languages' concepts should precisely describe what is in its focus, everything else should be omitted.

LANG-6 *Clarity:* The languages' concepts should be intuitively distinguishable by the modeler.

LANG-7 *Consistency:* The provided concepts should have a clear, unique semantics i.e., there should be no ambiguous interpretation of the model.

LANG-8 *Orthogonality:* Orthogonality of aspects of the reality should be reflected by different modeling concepts. Related aspects of the reality should be reflected by related concepts.

LANG-9 *Generality:* The concepts should be on a general level, i.e., not dependent on a certain application domain.

6.3 The MUVIEMOT Method

This section introduces MUVIEMOT - a modeling method aiming at supporting method engineers and tool developers in the conceptualization of multi-view modeling tools. The approach is specifically designed to capture the specifics of multi-view modeling methods. The goal of the approach is to guide the users in the development of a conceptual tool design, following a sequential procedure, constituting the MUVIEMOT lifecycle (see Figure 33).

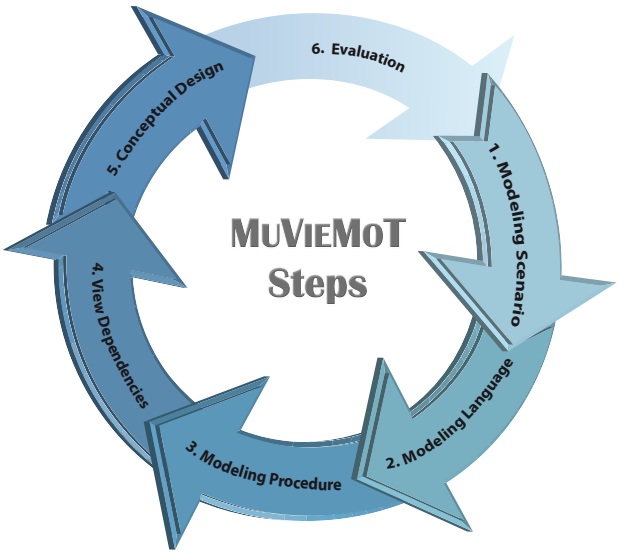


Figure 33: The MUVIEMOT life cycle

The research field of method engineering can be characterized as *"the engineering discipline to design, construct and adapt methods, techniques, and tools for the development of information systems"* (BRINKKEMPER, 1996, p. 276). Hence, a novel method, enabling the efficient development of a conceptual design for multi-view modeling tools, is introduced in the following. The method is not constrained by a certain development platform or programming language. *"At design time, we must articulate the software architecture and component functionalities that satisfy functional and non-functional requirements while avoiding unnecessary implementation constraints"* (KRAMER, 2007, p. 42). In this regard, the MUVIEMOT method aims to be on a generic level, however, the realized conceptual design can be instantiated to a certain development platform, e.g., by generating platform-specific meta models.

In the following, the MUVIEMOT method is introduced by describing the sequence of steps of the lifecycle illustrated in Figure 33. Each step, except of the evaluation step, is introduced

by describing its purpose and the modeling language, specifically designed to answer the purpose. Subsequently, section 6.3.7 outlines the integration of the different steps by means of the modeling procedure of MUVIEMOT.

6.3.1 Step I: Modeling Scenario

6.3.1.1 Purpose

In the first step of the approach, the goal is to explicate the modeling scenario the modeler is exposed to. More precisely, the purpose is to characterize the context in which a human modeler acts to create a multi-view model of an existing or postulated subarea of the real world. Thereby, the modeler is guided by the constituents of a modeling method and supported by a modeling tool. The role of the modeler within a modeling scenario is as follows (cf. (BORK AND SINZ, 2013, p. 29)): The modeler pursues certain goals which reflect the purpose of the model to be built. The goals refer both to the purpose of the model and the stakeholders to use the model. The modeler is guided by a metaphor, giving the general idea of understanding the modeling language and delimiting and reconstructing the relevant part of the real world.

The first step provides an overarching informal representation of the multi-view modeling method. No technical considerations are taken into account. Moreover, viewpoints are only described by means of their target audience and their relationships to meta models. This early stage of development might appear trivial or unnecessary, however it is a very crucial and important step. By explicating the modeling scenario, one is able to gather an overview of the complex setting of multi-view modeling. One major decision that needs to be made is which multi-view modeling principle (cf. section 5.4) the modeling tool should utilize. This decision can depend on the specification of the modeling method at hand or the goals of the modeling tool. In either case, the chosen principle has significant influence on the following MUVIEMOT steps.

6.3.1.2 Modeling Language

The modeling language, supporting the creation of a modeling scenario model is comprised of the following concepts: It relates the *modeler* to the *goals* by a *pursues* relationship, to the *metaphor* by a *guided by* relationship, to the *relevant subarea of the real world* by a *delimits* relationship, and most importantly to the *model* he or she creates according to a *multi-view modeling principle* and to the *viewpoints* by a *looks at* relationship. Finally, viewpoints are related to the *meta models* by means of a *projection* or *selection* relationship. The focus of the method engineer in this step should be on identifying all viewpoints and meta models on an informal level.

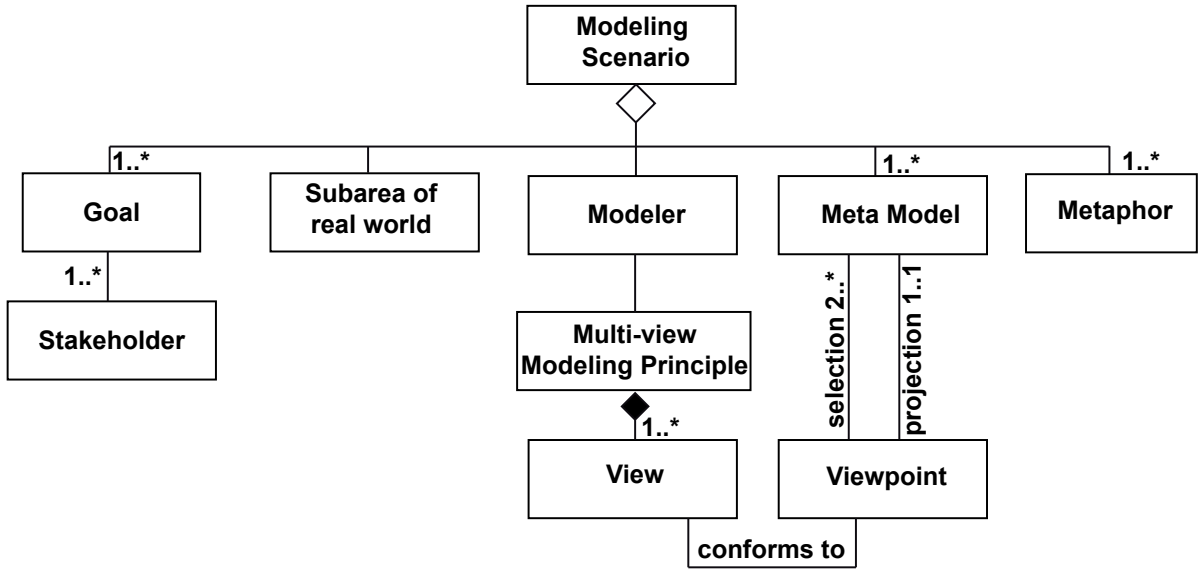


Figure 34: MUVIEMOT step I: Modeling Scenario

Figure 34 illustrates the constituents of the modeling scenario modeling language. Cardinalities between concepts are given in (min..max) notation. For readability reasons, cardinalities are only visualized if they are unlike 1..1. Hence, a modeling scenario is composed of one to many goals, a relevant subarea, a multi-view modeling principle, one to many meta models and one to many metaphors guiding the modeler. Goals are defined by one to many stakeholders. The modeler creates a model, composed of one to many views, according to a multi-view modeling principle. Each view conforms to exactly one viewpoint which in turn is related to exactly one meta model by a projection relationship, or, two to many meta models by a selection relationship, respectively.

6.3.2 Step II: Modeling Language

6.3.2.1 Purpose

In the second step of the MUVIEMOT method, the emphasis is on the specification of the viewpoints' modeling language. In the first step, the relationships between viewpoints and meta models have been introduced. Now, these relationships are being detailed by explicitly specifying the concepts of the viewpoints' modeling language by relationships to concepts of the (multiple) meta model(s). Hence, concepts are introduced in meta models first, before being reused in one to many viewpoint modeling languages. As a result of this step, meta models and viewpoint models are specified. More precisely, the modeling language step comprises the following sequence of tasks:

1. Define meta models
   Meta models play an important role in the MUVIEMOT method. Consequently, the definition of the meta models serves as the anchor for the specification of the viewpoints' modeling languages. Generally, two cases can be distinguished: (1) one integrated meta model, all viewpoints' modeling languages are derived from by a projection relationship; and (2) several loosely coupled meta models, e.g., an individual meta model for each viewpoint, all viewpoints are derived by a selection relationship.

2. Define viewpoints
   Except of the modeling concepts, derived from the meta models, each viewpoint also comprises a set of attributes: *name*, a meaningful name; *stakeholders*, the target audience; *goals*, the pursued goals; *relationships*, connecting the modeling concepts originating from different meta models with each other; *constraints*, constraining the set of valid relationships between concepts of a viewpoint.

3. Define viewpoint-specific attributes
   The modeling concepts, derived from the meta models they have been specified in, come with a predefined set of attributes, e.g., name, visualization. However, viewpoint-specific attributes may be added and derived attributes may be adopted to the viewpoint: This enables e.g., a viewpoint-dependent visualization of concepts.

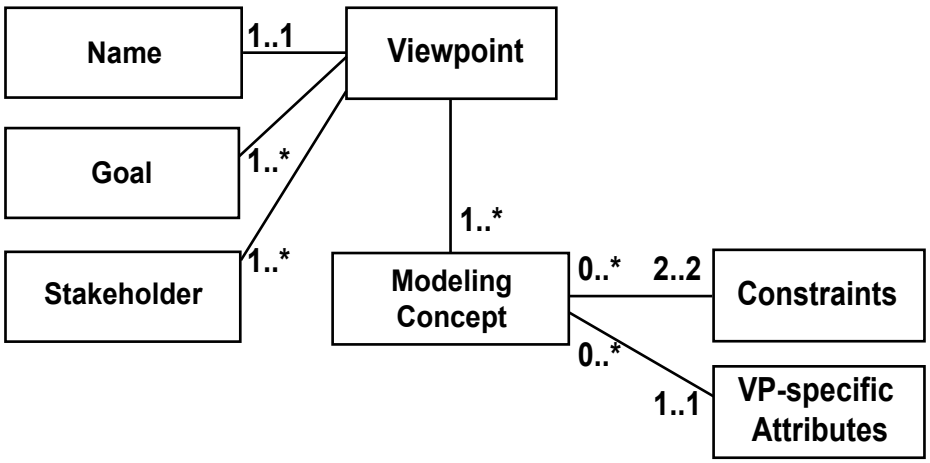


Figure 35: MUVIEMOT step II: Modeling Language

6.3.2.2 Modeling Language

Figure 35 visualizes the constituents of the viewpoint modeling language as part of the second step of MUVIEMOT. A viewpoint is defined by a unique name, one to many goals pursued and

one to many stakeholders addressed. Moreover, from a modeling perspective, the modeling concepts and optionally the constraints and viewpoint-specific attributes are part of the viewpoint specification. The meta model specification is not visualized here due to its simplicity. Meta models are composed of modeling concepts, either objects or relations, described by attributes, and constraints defining the set of valid relationships between objects and relations.

6.3.3 Step III: Modeling Procedure

6.3.3.1 Purpose

As emphasized already, specifying the modeling procedure for multi-view modeling methods is vital for the efficient implementation of a corresponding modeling tool (cf. (LANKHORST, 2009, p. 164f.)). However, *"relations between domains (views) is poorly defined, and the models created in different views are not further integrated"* (LANKHORST, 2009, p. 43). The aim of this step of the MUVIEMOT method is therefore to specify *multi-view modeling use cases* (cf. (BORK AND SINZ, 2013, p. 29ff.)). These special kind of UML use cases specify interactions between the modeler and viewpoints. Conventional UML use case diagrams (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 597ff.) are adopted by omitting the actor (as the actor is always the modeler) and relating all use cases to viewpoints. The method engineer distinguishes for each use case, whether it can be *triggered* in a certain viewpoint and whether executing the use case has an *effect* or a *conditionally effect* on a viewpoint. This needs to be decided for each viewpoint independently. Conventional use case relationships, e.g., *include*, *extend*, and *uses*, can be also modeled in order to capture relationships between functionality of the modeling tool.

The decision of the *Modeling Scenario* step according to the multi-view modeling principle has a significant influence on the effort of specifying the modeling procedure. If the *multi-view by generation* principle should be utilized, no propagation of modeling actions, performed in one view, must be specified. In this case, only transformations between complete viewpoints must be considered (i.e., *State Translations*, cf. section 5.4.2). By contrast, following the *multi-view by design* principle, dependencies between the views are already defined on the meta model level. These dependencies need to be transformed into corresponding consistency mechanisms (i.e., *Transition Translations*, cf. section 5.4.1).

6.3.3.2 Modeling Language: A UML profile for Multi-View Modeling Use Cases

In the *Modeling Procedure* step of MUVIEMOT, Multi-View Modeling Use Cases (MVMUCs) are constructed. MVMUCs are introduced by means of an extension of UML Use Case diagrams (OBJECT MANAGEMENT GROUP (OMG), 2011d), i.e., a UML Profile. Manifold UML extensions have been published recently, in order to combine the benefits of a standard-

conforming modeling language (e.g., experienced users, available tools) with the potential of a domain-specific modeling language (cf. (FONTOURA ET AL., 2001; SELIC, 2007; SCHAMAI ET AL., 2009).

UML use cases are defined as *"a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do"* (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 597). Central elements of use case diagrams are *actors*, *use cases*, and *relationships* between use cases. Actors are used to model entities outside the modeled system, interacting with the system by means of use cases. Actors can be used to model human beings, but also computer systems or roles, realized by an entity. A use case, *"is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system"* (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 606). The OMG standard defines four central relationships between use cases, the *extend relationship* the *include relationship*, and the *association* and *generalization relationship* inherited from UML class diagrams. Using the extend relationship, enables the specification of how the behavior of an extended use case may be included into the behavior of an extending use case. The extended use case should be independently form the extending use case. Hence, the extending use case may define some optional behavior, e.g., an extended use case *pay purchase at the pay desk* and an extending use case *request receipt*. The include relationship defines a direct dependency between two use cases. An including use case includes the behavior defined in an included use case, e.g., an including use case *pay by credit card* and an included use case *check credibility status*.

The conventional UML use cases have proven to be useful for the specification of the behavior of a system. Because of the manageable set of modeling concepts and their semantics, creating use case diagrams for a system is straightforward. However, in the complex setting of a multi-view modeling method, these conventional use case diagrams are not suitable. Therefore, extensions to UML use cases are introduced in the following. The UML provides a predefined set of mechanisms, one can use in order to extend and/or customize the syntax and the semantics of the UML modeling language. Such *UML Profiles*, have been created in a large number of projects[27] (ALDAWUD ET AL., 2003; FUENTES-FERNÁNDEZ AND VALLECILLO-MORENO, 2004; LUJÁN-MORA ET AL., 2006). In the following, a UML Profile is introduced that incorporates the benefits of intuitive use case diagrams with the specifics of the multi-view modeling domain. This aim is achieved by extending the UML meta model using the defined extension points, specifying the syntax and semantics for the extensions (cf. FUENTES ET AL. (2002)). A Multi-View Modeling Use Case compromises the following constituents (cf. (BORK AND SINZ, 2013)):

[27] see `http://www.omg.org/spec/`, last checked: 2013-08-22, for a selection of UML Profile specifications.

Actor In multi-view modeling use case diagrams, there is only one actor, the modeler. For readability reasons, we therefore suggest to omit the actor in the models.

Use Case A unique identifier (e.g., a number or name) followed by a meaningful and short description for the use case, depicting a certain system behavior.

Triggered In A set of viewpoints (zero to many), the related use cases can be triggered in. Zero viewpoints, if a certain use case can be triggered without being assigned to a certain view, e.g., directly on the multi-view model like a *create new multi-view model* functionality. By contrast, most use cases must be constrained to one or several viewpoints they can be triggered in.

Effect On For each viewpoint it must be decided, whether the execution of an use case has always (value *"Yes"*), conditionally (value *"Cond"*) or never an effect on that viewpoint. Conditionally effects on viewpoints are of particular interest as they challenge for both, identification and implementation.

References A list of references to use cases by means of the *include*, *extend* or *generalization* relationship. The references are defined by the relationship type and the identifier or number of the related use case.

Multi-View Modeling Use Cases, as introduced above, can be visualized using a tabular approach with all constituents as columns and the use cases as rows (see Table 14).

Table 14: Tabular specification of Mutli-View Modeling Use Cases (cf. (BORK AND SINZ, 2013)

Use Case	Triggered In			Effect On			References
	VP 1	VP 2	VP 3	VP 1	VP 2	VP 3	
1. Use Case 1	Yes	Yes		Cond.	Yes		include(3)
2. Use Case 2					Cond.		extend(1)
3. Use Case 3			Yes				include(2)

However, using graphical representations fosters structuring of information in a more appropriate way and intuitive understanding by human beings. Therefore, a graphical visualization of

MVMUCs is emphasized. The alignment of the elements is as follows: On the left border are all viewpoints depicted that can be related to use cases by means of *triggered-in* relationships. On the right border, all viewpoints are visualized again, but now only those related to use cases by means of *effect-on* or *conditionally effect-on* relationships. In the center, conventional use cases together with conventional use case relationships are visualized. Figure 36 illustrates a sketch of a graphical representation of multi-view modeling use cases (for a better comparison, the same data is used as in Table 14).

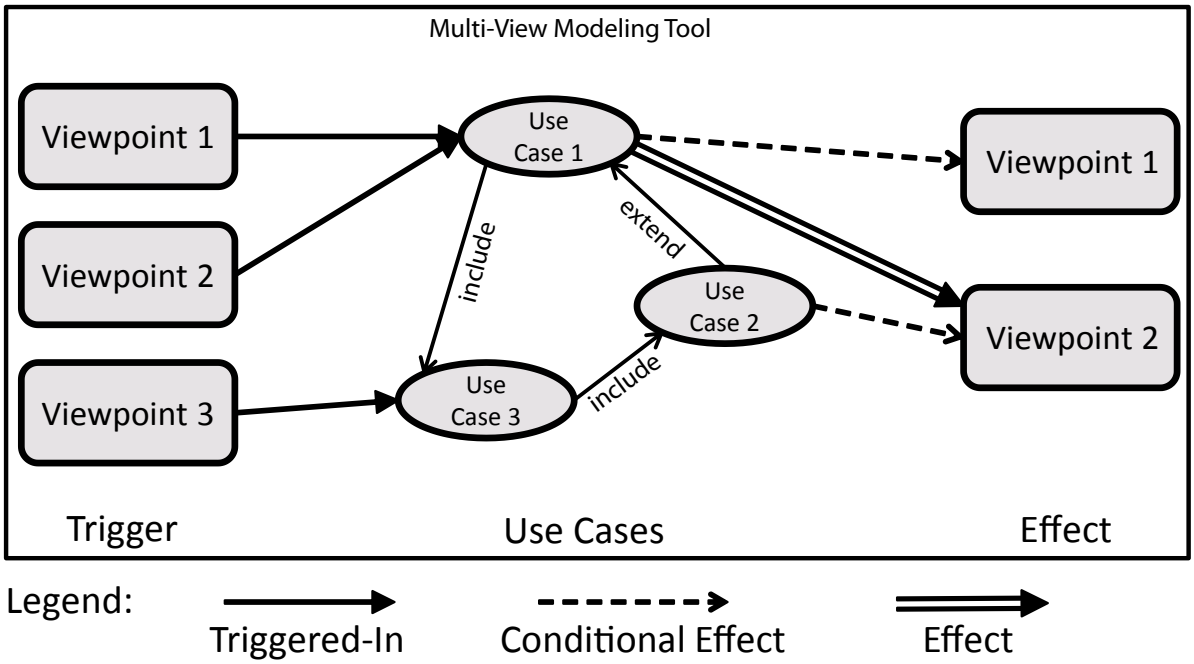


Figure 36: Sketch for a graphical specification of Multi-View Modeling Use Cases

The goal of the *Modeling Procedure* step of MUVIEMOT is to capture the interactions between the modeler and the multi-view modeling tool in an intuitive and efficient but also formalized way. The proposed language combines the benefits of a commonly used standard for system's behavior specification with specifically designed concepts from the multi-view modeling domain.

6.3.4 Step IV: Viewpoint Dependencies

6.3.4.1 Purpose

Consistency management is of major importance for acceptance and efficiency of the multi-view modeling tools (cf. section 2.2.4 for a detailed introduction into consistency management). However, a prerequisite for consistency managing is to be aware of all relationships between concepts of the multiple views - the viewpoint dependencies. As introduced earlier, syntactic

and semantic dependencies can be distinguished. The former can be derived by comparing the meta models of viewpoints and identifying overlapping areas, i.e., modeling concepts captured in multiple viewpoints. By contrast, identifying semantic relationships is not as obvious. A deep understanding of the semantics of different viewpoints needs to be given in order to identify these dependencies. Hence, viewpoint dependencies should always be specified by the method expert (or method owner) and not by tool developers.

The aim of this MUVIEMOT step is to provide an intuitive modeling language, designed to enable the specification of viewpoint dependencies in a model-driven manner. Choosing a graphical modeling language instead of a more formalized notation, e.g., the Object Constraint Language (OCL), was motivated by the intended creator the the viewpoint dependencies - the method expert. Hence, dependencies should be specified on an abstract level, considering the viewpoint meta models, but without considering technical implementations.

6.3.4.2 Modeling Language

The *Viewpoint Dependencies* modeling language enables the specification of consistency constraints by relating modeling concepts to the viewpoints they are considered in. Moreover, the method expert 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.

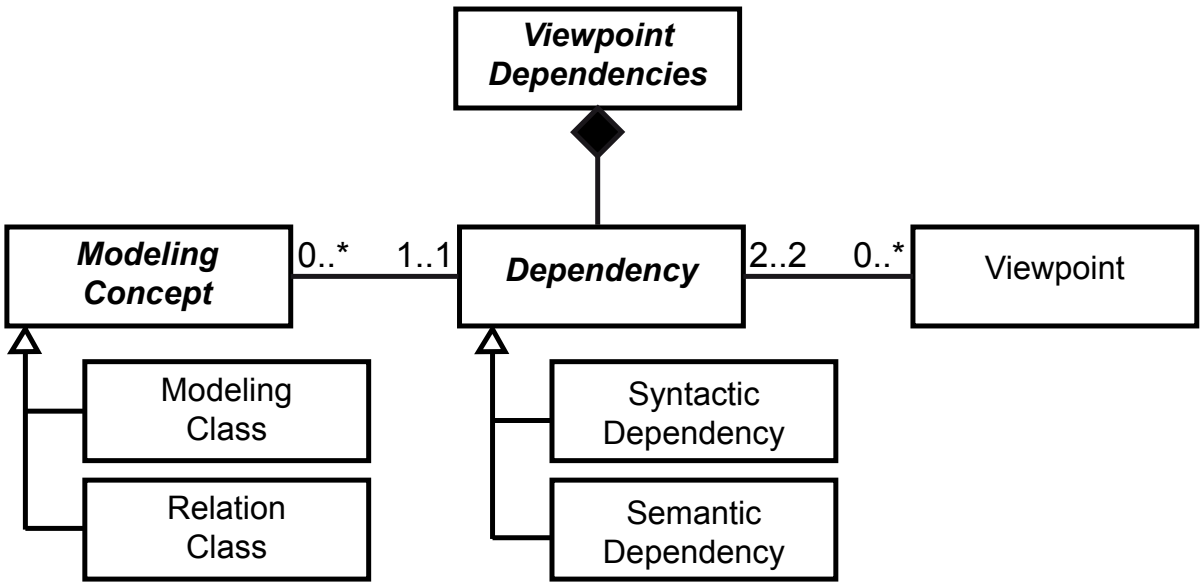


Figure 37: MUVIEMOT step IV: Viewpoint Dependencies

Figure 37 illustrates the constituents of the *Viewpoint Dependencies* modeling language. A

dependency, either of syntactic or semantic nature, is related to exactly one *modeling concept*, either a modeling class or a relation class, on the one side, and exactly two viewpoints on the other. Both, viewpoints and modeling concepts can be part of none or multiple dependencies.

Note, that the detailed specification of the dependency is not covered in the graphical model. The precise specification of the modeling concepts and attributes that need to be kept consistent is part of the *Conceptual Design* step.

6.3.5 Step V: Conceptual Design

6.3.5.1 Purpose

The fifth step of the MUVIEMOT method comprises the information specified in the preceding steps into an overarching specification of a multi-view modeling tool, referred to as conceptual design. In general, conceptual design specifications can be developed in two levels of abstraction: First, on a conceptual level and independent of development technologies like tool development platforms or programming languages. Second, the constituents of the conceptual design are specifically aligned to the functionality and components of a development technology. The former can be achieved with knowledge on the multi-view modeling method at hand, whereas the latter also requires significant knowledge on the development technology. In total, a conceptual design is built on three building blocks: **Functional Requirements**, **Consistency Requirements**, and **Non-Functional Requirements**.

Independently of the abstraction level, the conceptual design is aiming at providing more precise **functional requirements**. Therefore, the incomplete or ambiguous information of e.g., the modeling procedure step is now refined. The *effects* and *conditional effects* modeled in the multi-view modeling use cases are now precisely specified using e.g., natural language or a more formalized notation. For each use case defined in the third MUVIEMOT step, its effect on viewpoints and consistency issues are specified. Especially in the context of a multi-view modeling method, even rather simple add, remove or modify operations can force the developer to implement a set of more or less complex operations potentially applied to several views in order to preserve a consistent state of the model. An emphasis of the specification should be also on the conditional effects as they are not obvious and it is not likely that tool developers, being generally no method experts, would identify those conditional effects by themselves.

Another building block of the conceptual design are **consistency requirements**. As mentioned already, the *viewpoint dependencies*, identified in the preceding step, are not specified in detail, yet. The conceptual design, uses these dependencies and refines them. Consequently, it must be specified for each dependency, which attributes are concerned by a change, and, depending on the multi-view modeling principle, when to execute the consistency mechanisms. *State translations* and/or *transition translations* (cf. section 5.4) need to be specified in an

appropriate and unambiguous way.

The conceptual design of a modeling tool should also consider **non-functional requirements** (NFRs). In addition to classical NFRs like intuitivity, efficiency, stability, an emphasis should be on the general way of carrying multi-view modeling and the usage of the tool by a human modeler.

The aim of the conceptual design is to capture all functional and non-functional requirements of a multi-view modeling tool. The input of all preceding steps is combined and integrated into an overarching tool specification with an emphasis on consistency. The conceptual design specification should provide an unambiguous development handbook, enabling tool developers to implement a consistency-preserving multi-view modeling tool more efficiently.

6.3.5.2 Modeling Language

Figure 38 illustrates the constituents of a *conceptual design* model (cf. (BORK AND SINZ, 2013, p. 31f.)). A conceptual design is composed of *functional requirements*, *non-functional requirements*, and *consistency requirements*. The latter two are specified informally, e.g., using natural language. Functional requirements are specified in a semi-formal way by providing the modeler with a structured set of properties. These properties enable comprehensive specification of the tool's functionality with respect to the multi-view scenario.

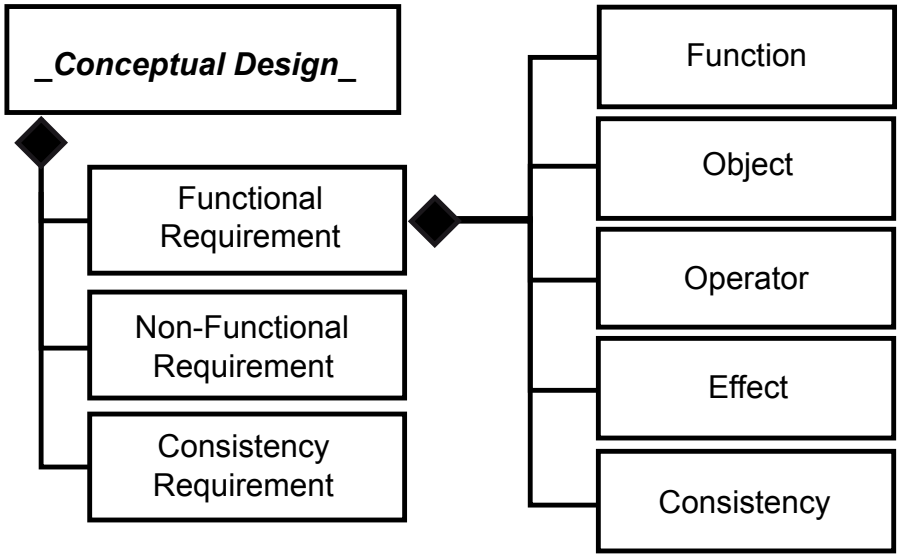


Figure 38: MUVIEMOT step V: Conceptual Design

The notation of the conceptual design model is twofold: First, tabular representations are realized as they have proven their efficiency in requirements specifications (ZHU ET AL., 2014). A benefit of using tables is the intuitive interpretation by the modeler and the inherent structuring for huge amounts of data. Moreover, the modeler is provided a graphical visualization.

The specification of a functional requirement in the conceptual design step of MUVIEMOT comprises the following information (cf. (BORK AND SINZ, 2013, p. 32)):

Function Denomination of the functionality by means of an use case of the modeling method, specified in the modeling procedure step, e.g., *add model element*, *delete model element*, *generate model*; or functionality of the modeling tool, e.g., layout algorithms, model management. If derived from a multi-view modeling use case, the name of the use case can be used as denomination of the function.

Object Every functionality of the multi-view modeling tool should be related to an element, or a set of elements, of the viewpoints' meta models. The relationship indicates, that a certain functionality can be triggered at e.g., a certain model element or a set of viewpoints. Some functionality may be related to components of the modeling tool, e.g., context menu, tool window, model window; or the model as a whole, e.g., changing the font size of the labels or changing the read/write mode of a viewpoint.

Operator Signature of the operator by means of the explicit name, the functionality should have in a corresponding modeling tool. There is a $1..^*$ relationship between a function and an operator. This enables composition of functionality that semantically corresponds to each other, e.g., the function *split sequence flow* of a process model can be decomposed into multiple operators, e.g., *add 'OR' split* and *add 'XOR' split*. Furthermore, this hierarchical classification enables structuring of all modeling operators.

Effect The description covers all effects caused by the execution of the operator on all viewpoints. It can be in several degrees of formalization, i.e., a pure informal, textual description of the effects, a semi-formal description using pseudo-code notation, or a formal specification of the operator by using a programming language or mathematical notation.

Consistency The execution of modeling operators in a multi-view setting always puts the consistency of the views at risk. Description of the possible inconsistencies is therefore vital for ensuring consistency-preserving modeling. The multi-view modeling use cases act as an initial information source. The identified conditional effects need to be specified thoroughly in the conceptual design. Hence, the conditions of an effect on a certain viewpoint need to be investigated. Usually, the description is on an informal level. However using OCL constraints or formal specification languages can be also utilized - depending on the experience of the modeler.

The conceptual design specification is usually consolidated in a tabular description. As the conceptual design can be performed on an informal level using natural language, the notation does not need to be more formal. The resulting requirements specification should enable a tool

developer, not necessarily a method expert, to implement a consistency-preserving multi-view modeling tool.

6.3.6 Step VI: Evaluation

6.3.6.1 Purpose

As the MUVIEMOT lifecycle (see Figure 33) indicates, an iterative development approach is promoted. Each iteration results in a comprehensive requirements specification of a multi-view modeling tool that needs to be evaluated according to the modeling method at hand. Generally, evaluation can be performed by discussing the outcomes with method experts informally. Moreover, the specification can be transformed into a prototypical tool implementation. This prototype can be given to method experts and potential users in order to gain feedback.

6.3.6.2 Evaluation Techniques

MUVIEMOT does not specify the evaluation technique, however, some possible evaluation techniques from the design-science research area are applicable (cf. (PEFFERS ET AL., 2012, p. 402)): **Expert Evaluation**, *"Assessment of an artifact by one or more experts"*; **Prototype**, *"Implementation of an artifact aimed at demonstrating the utility or suitability of the artifact"*; **Case Study**, *Application of an artifact to a real-world situation, evaluating its effect on the real-world situation*; and **Illustrative Scenario**, *Application of an artifact to a synthetic or real-world situation aimed at illustrating suitability or utility of the artifact*.

6.3.7 The MUVIEMOT Modeling Procedure

The MUVIEMOT method promotes a sequential and iterative approach towards conceptual design specification. Figure 39 illustrates the MUVIEMOT modeling procedure. In each step of the approach, the modeler can go back to the preceding step if he/she figures out that something is missing or specified incorrectly. After one step is concluded, the modeler moves one step ahead. The results gained from the evaluation of the conceptual design then indicate whether or not an additional iteration is required.

Depending on the results gained in the evaluation step, the modeler is able to jump into a certain step of the method, and, starting from this step, consecutively changing the models of the succeeding steps accordingly. Hence, if some functionality is missing, the modeler does not need to start with the modeling scenario. Instead, he or she can start by adding the functionality to the multi-view modeling use cases. All succeeding models can be updated accordingly in order to include the new use case.

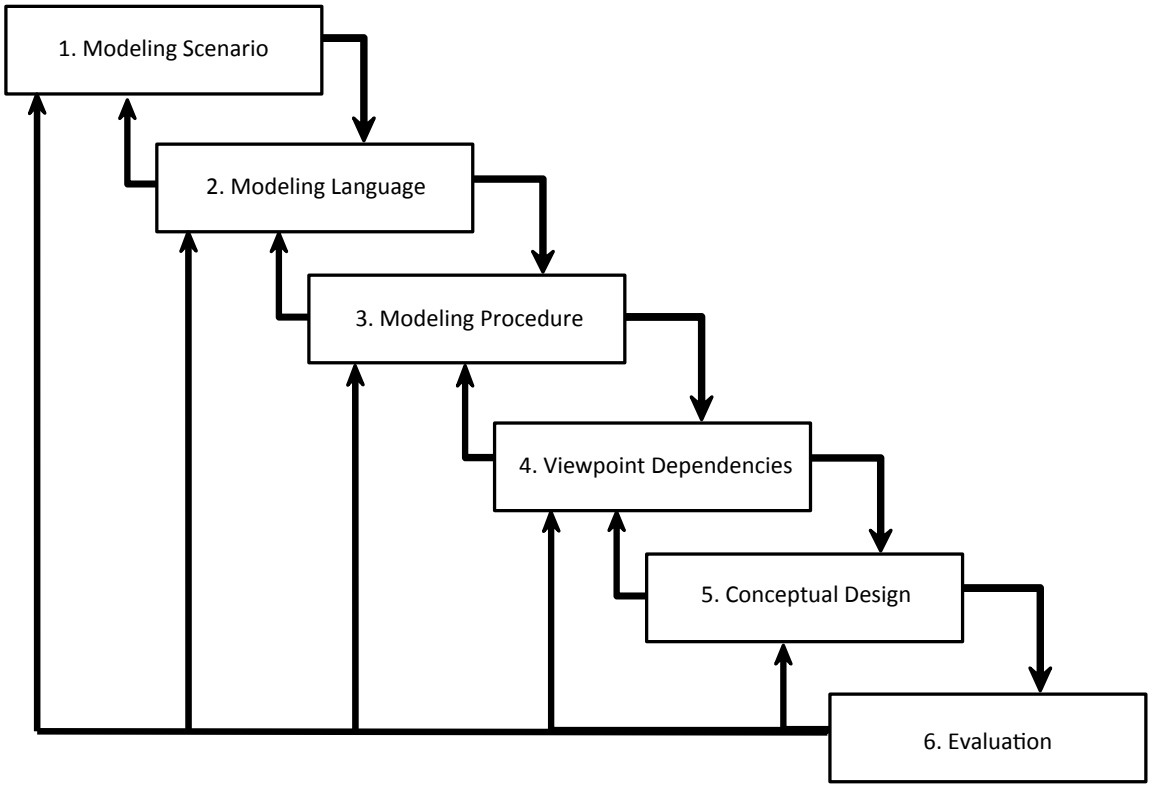


Figure 39: The MUVIEMOT modeling procedure

6.3.8 Integration of the MUVIEMOT steps

Except the evaluation, all MUVIEMOT steps, as introduced earlier, utilize specifically designed modeling languages. The different modeling steps of the approach are coupled by means of relationships between modeling languages of different steps and conceptual transformations between different steps.

Figure 40 illustrates the different modeling languages of the MUVIEMOT method in a compact and integrated manner. The illustration adopts UML class diagram notation with Min..Max notation of the relationship cardinalities whenever it is not a 1..1 cardinality. Several relationships between the different MUVIEMOT steps are visualized by means of red dotted lines between concepts of the corresponding modeling languages. For readability reasons, not all relationships are depicted in the illustration.

These 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 potential for model transformations between MUVIEMOT steps. For example, when creating a *Multi-View Modeling Use Case* model, all viewpoints can be derived from the already defined *Modeling Language* step. Also, an initial set of functional requirements of the *Conceptual Design* model can be derived from the use cases specified in the *Modeling Procedure* step.

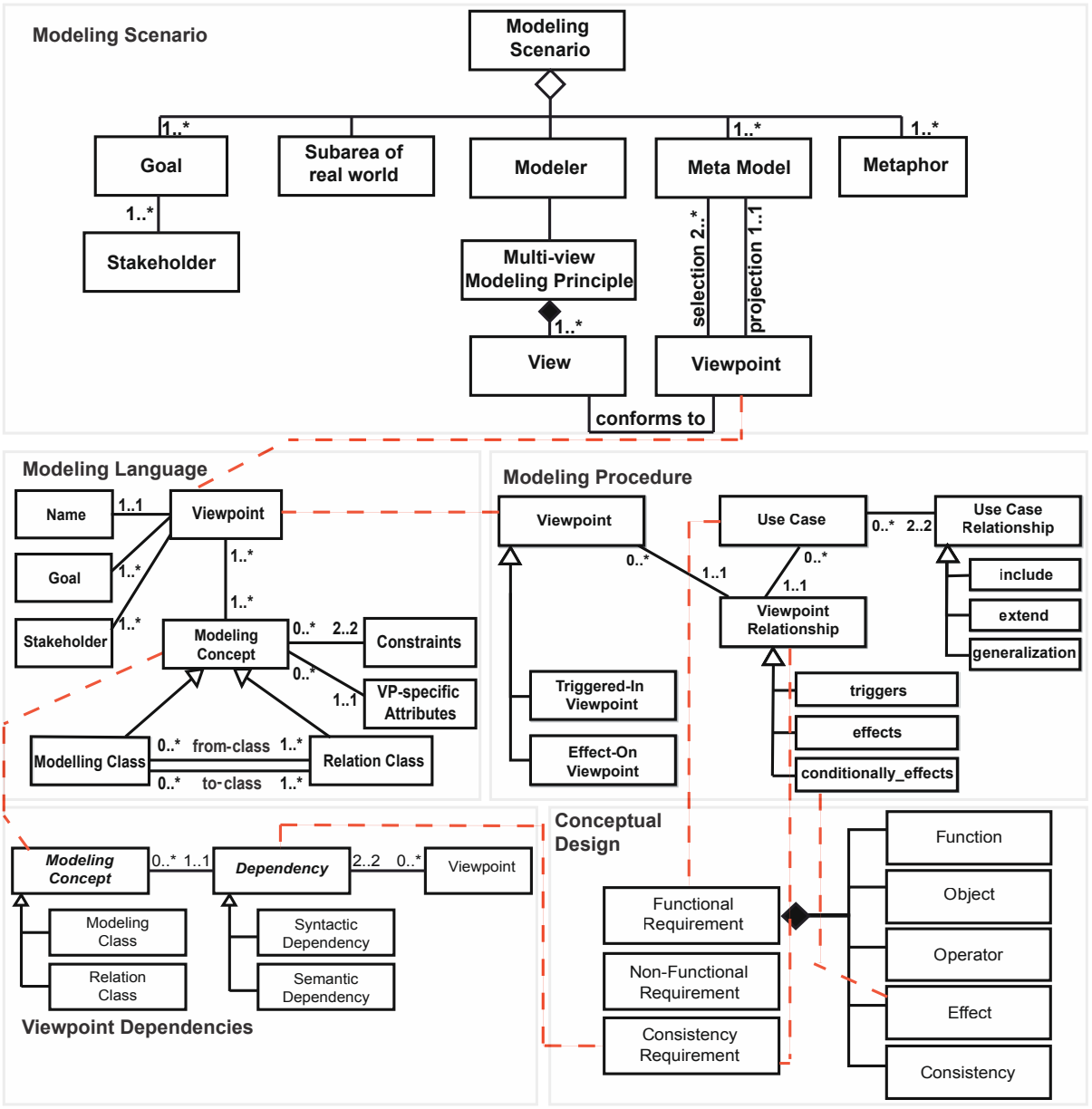


Figure 40: The MUVIEMOT meta model overview

In the current version, the different steps of the approach need to be cognitively integrated by the modeler. However, the specification of the integration by means of relationships between modeling languages eases the development of a modeling tool for the MUVIEMOT method (see section 8).

6.4 Evaluation

The MUVIEMOT method can be considered an artifact, realized as a result of conducting design science research. As PEFFERS ET AL. (2012) emphasize, illustrative scenarios are an adequate

and widely adopted evaluation technique for artifacts of type method. The illustrative scenario evaluates the artifact according to suitability and utility. Moreover, evaluation can be performed by logical arguments and expert evaluations (PEFFERS ET AL., 2012, p. 402). Consequently, the evaluation of the MUVIEMOT method is based on the application of independent evaluation techniques: In section 6.4.1, the requirements identified in section 6.2 are compared to the constituents of the method. Moreover, section 6.4.2 discusses the method in the context of the relevant research questions and a comprehensive SWOT analysis is performed (section 6.4.3). Finally, in section 7, the conceptual design of a multi-view modeling tool for the SOM method is created by means of utilizing MUVIEMOT in a comprehensive *illustrative scenario* (PEFFERS ET AL., 2012).

6.4.1 Requirements Analysis

Using the requirements specified in section 6.2 a first evaluation of the proposed method is performed. Thereby, each identified requirement is evaluated against the current status of the MUVIEMOT method. Table 15 summarizes the requirements evaluation in a tabular notation. Each requirement is analyzed by the criteria: id of the requirement [ID]; whether this requirement is fulfilled completely (●), partly (◐), or not yet (○) [Value]; and a description of how MUVIEMOT targets this requirement [Description].

Table 15: MUVIEMOT method requirements evaluation

ID	Value	Description
		Literature Requirements
LIT-1	●	MUVIEMOT provides specialized modeling languages, used in the *Modeling Language* step, to specify the syntax, semantics, and notation of viewpoints.
LIT-2	●	Relationships between viewpoints can be specified in the *Viewpoint Dependencies* step of MUVIEMOT.
LIT-3	●	Due to the separation between meta models and viewpoint models in MUVIEMOT, the modeler can either define a common notation for a modeling concept in the *Meta Model* model, or he/she can define a viewpoint-specific notation in the corresponding *Viewpoint* model.

LIT-4	●	All modeling languages of the MUVIEMOT steps utilize abstract concepts specific for the application domain. No general-purpose or already existing modeling language is used as it won't address the specific needs.
LIT-5	◐	All visualizations of the MUVIEMOT method aim to be intuitive. However, multiple instantiations of the method by different and independent users need to be realized in order to elaborate on the quality of the current visualizations and determine possible future improvements.
LIT-6	●	*Modeling Procedure* step and *Conceptual Design* step are designed with two alternative notations, e.g., in the former case a graphical notation using *Multi-View Modeling Use Cases*, and a tabular notation are provided. The conceptual design specification utilizes also a graphical notation for rudimentary covering the requirements, and a tabular notation for the more detailed specification.
LIT-7	●	The *Modeling Language* step of MUVIEMOT utilizes meta modeling as a means of specifying the viewpoints' modeling language in a formalized manner. Moreover, by utilizing modeling languages, all modeling steps enable at least a semi-formal specification.
LIT-8	◐	The relationships between viewpoints and meta models can be defined by a *projection* or *selection* relationship in the *Modeling Scenario*. The general way of carrying out multi-view modeling can be specified in an informal way. However, the decisions determine the need for specifying certain synchronization mechanisms.
LIT-9	●	The MUVIEMOT method supports the modeler in specifying synchronization mechanisms for projective and selective multi-view modeling. In the former case, viewpoint modeling language models and meta models enable the automatic identification of syntactic dependencies. Semantic dependencies can be added manually to cover the latter case. The dependencies can be defined in multiple levels of detail.

LIT-10	○	The specification of hierarchical relationships between viewpoints is not supported by the method, yet. Besides one approach found in literature, no other approach utilizes this functionality. Nevertheless, viewpoint navigation seems to be a powerful weapon for sustaining an overview when the number of views increases but a meaningful viewpoint hierarchy can be established.
LIT-11	◐	At the time of evaluating MUVIEMOT, neither ontological completeness nor ontological clarity have been violated. Notably, especially in the former case, a formal prove is not possible, yet.
		Implementation Experience Requirements
EXP-1	◐	Currently, the *Viewpoint Dependencies* model supports also the specification of model transformations between different viewpoints. However, the detail of specification does not cover the manifold possibilities of transformations between modeling languages, yet. However, by relating the dependencies on the viewpoints' meta models forces the modeler to be precise instead of using uncontrolled natural language descriptions.
EXP-2	●	An emphasis of MUVIEMOT is on the specification of the interactions between the modeler and the multi-view modeling tool. The *Multi-View Modeling Use Case* model eases the mapping between modeling operations and viewpoints. The method engineer is forced to decide for each viewpoint and modeling operation on its triggers and effects.
EXP-3	●	In *Multi-View Modeling Use Case* models, use cases and viewpoints are related by *triggers*, *effects*, or *conditionally_effects* relationships. Moreover, the effect relationships are further specified in the *Conceptual Design* step. In this regard, not only the existence but also the specificity of an effect is considered.
EXP-4	◐	The conceptual design can be considered as a specification of tool functionality like consistency checks, analysis, and viewpoint transformation functionality. It eases the development of complementary tool functionality not considered in the core of the multi-view method.

	Usability Requirements	
USE-1	●	The MUVIEMOT method is decomposed into a procedure of six steps. Hence, the complexity of designing a conceptual design for multi-view modeling tools is decomposed into several, manageable steps. The focus moves continually from a conceptual and mereological abstraction level towards a technical tool development abstraction level.
USE-2	●	As the MUVIEMOT modeling procedure (cf. Figure 39) indicates, backtracks between steps are supported. Moreover, the complete approach utilizes an iterative procedure enabling the modeler to adopt the models if deficiencies have been identified during the evaluation.
USE-3	◐	Although trying to provide an intuitive notation for the different MUVIEMOT modeling languages, there is potential left for further improvements in future versions.
	Modeling Language Requirements	
LAN-1	●	The conceptual design specification is decomposed into multiple steps, each focusing only on a subset of the multi-view modeling method. The modeling language of MUVIEMOT is accordingly decomposed along these steps.
LAN-2	●	Sections 6.3.1 until 6.3.6 entirely describe the steps of the MUVIEMOT method. Table 16 furthermore aligns the steps to different user roles, specifying the objectives, input and output of each step.
LAN-3	●	The different modeling steps are not only semantically but also syntactically integrated by e.g., using the viewpoints specified in the *Modeling Scenario* step in the *Modeling Language*, *Modeling Procedure*, and *Viewpoint Dependencies* step. Moreover, the method emphasizes on the transformation of *Multi-View Modeling Use Cases* into functional requirements, depicted in the *Conceptual Design* model. Figure 40 provides an overview on the integration of the different method steps.

LAN-4	◐	Modeling language completeness is, as already stated for ontological completeness, impossible to guarantee. However, while using the method in multiple illustrative scenarios, no missing concepts could be identified. It should be noted, that completeness is dependent on the identified relevant aspects and the pursued goals.
LAN-5	●	For each concept of the different modeling languages a precise specification of its purpose is given to the modeler.
LAN-6	●	Clarity of modeling language concepts is supported by the developed notation. Thus, enabling intuitive understanding and utilization of the modeling languages.
LAN-7	●	Non-ambiguous semantics of modeling language concepts has been an important requirement during the language development.
LAN-8	●	The orthogonality language requirement has been considered during the development of the different MUVIEMOT modeling languages. e.g., *Modeling Classes* are distinguished from *Relation Classes* in the *Modeling Language* step, however they can be related to each other. By contrast, the viewpoint concept is used in multiple steps of the approach.
LAN-9	●	The concepts of the MUVIEMOT modeling languages have been emerged during multiple applications of the method. Hence, it is guaranteed, that the concepts are not dependent on a certain multi-view modeling method. By contrast, the method is aimed to be on a generic level, allowing for application with arbitrary multi-view modeling methods.

Legend: ● $\widehat{=}$ completely fulfilled, ◐ $\widehat{=}$ partially fulfilled, ○ $\widehat{=}$ not fulfilled

As Table 15 indicates, not all requirements are completely fulfilled. Some of them are left for future research, some did not seem feasible during a PhD thesis. The focus was on capturing all commonly present and vital aspects of multi-view modeling methods in MUVIEMOT. Future versions of the method should consider incorporating also rare and unique aspects of multi-view modeling methods like viewpoint hierarchies and derived viewpoint navigation functionality.

6.4.2 Comparing MUVIEMOT to the Research Questions

This section is aimed to provide a comparison of the research questions specified in section 1.2 and the constituents of the MUVIEMOT method.

RQ I: What is the theoretical and conceptual foundation of multi-view modeling and multi-view modeling methods?

The theoretical and conceptual foundations of multi-view modeling have been discussed in detail in section 2 and section 5. These foundations have had a significant influence on designing the constituents of the MUVIEMOT method and the corresponding modeling languages. In this regard, theoretical considerations on the different ways of carrying out multi-view modeling, the different relationship types between meta models and viewpoint models, or the different types of viewpoint dependencies are reflected and codified in the modeling languages of MUVIEMOT.

RQ II: Which consistency issues must be considered in the context of a multi-view modeling method?

Different classes of consistency issues raised by the interaction of human beings with multiple views have been discussed in this thesis. Horizontal, vertical, intra- and inter-model consistency have been established along with an understanding of model conformity and validity. These investigations foster a common understanding and realize an awareness towards the manifold consistency issues that need to be discussed in the context of designing a multi-view modeling tool.

RQ III: What influence does the availability of an integrated meta model have on the conceptual design of a multi-view modeling tool?

Section 5.3 investigated different ways of integrating viewpoints by means of their relationships to one or more meta models. The two identified approaches have been reflected and codified in the MUVIEMOT modeling scenario model by a projection and selection relationship that can be used in order to define the semantic relationship between the viewpoint and meta model at hand. This thesis, with the SOM method as illustrative scenario, focuses on the integrated meta model approach. However, the method is prepared for its usage in scenarios where several meta models need to be integrated first.

RQ IV: How can the conceptual design specification of multi-view modeling tools be supported by a methodical approach?

The result of this investigation is the MUVIEMOT method aiming at bridging the gap between method experts on the one side and tool developers on the other. The

method proposes a sequential approach, enabling to handle the complexity of modeling tool conceptualization into manageable steps. Every step of the approach utilizes a conceptual modeling language that provides the concepts specific for the intended purpose and audience.

RQ V: What are the benefits of formalized modeling method specifications?
Section 4 thoroughly discussed the possibilities and benefits of a formalized modeling method specification by investigating a set of commonly used enterprise modeling methods. The results emphasized the specification of the MUVIEMOT method not only on a conceptual and informal level but also on a formalized level wherever appropriate, e.g., by providing meta models for the modeling languages, sample models for the notation, and a modeling procedure.

RQ VI: How can a graphical modeling language and a corresponding modeling tool ease the application of the method?
The utility of MUVIEMOT has been further increased by developing a modeling tool and using this tool in an illustrative scenario (see section 8). The tool not only transformed the meta models into graphical editors, it also enabled transformation of models in adjacent MUVIEMOT steps, therefore fostering reuse; and the transformation of the comprehensive conceptual design into an initial implementation of a multi-view modeling tool, therefore enabling model-driven development.

6.4.3 SWOT Analysis

In the following, a comprehensive discussion on the quality of the current state of the MUVIEMOT method is performed by means of a SWOT analysis. Major strengths, weaknesses, opportunities and threats are briefly discussed in order to evaluate the method.

Strengths Major strengths of the approach originate from its domain-specificity. All modeling languages are specifically designed according to the purpose they should fulfill. The MUVIEMOT method enables the efficient and comprehensive specification of a conceptual design for a multi-view modeling tool.

Providing a model-driven approach also improves the efficiency of developing a comprehensive conceptual design. Compared to unstructured requirements specification using natural language, the utilization of the modeling languages fosters efficiency of the development process and comprehensiveness of the specification. The graphical visualization of the MUVIEMOT steps, e.g., the *Viewpoint Dependencies* step, also helps to identify and document the dependencies between the viewpoints more easily.

The modeling procedure provides guidance for the modeler. Hereby moving from an informal and abstract perspective on the multi-view modeling tool in the *Modeling Scenario* towards a concrete tool specification in the *Conceptual Design* in a stepwise manner.

The semi-formally specified modeling languages for all steps of the method ease the utilization by modelers and the codification of the knowledge by means of a multi-view modeling tool.

Weaknesses In some steps of the MUVIEMOT method, only informal or semi-formal specifications are produced. As section 4 elaborated, a formalized specification is superior to informal or semi-formal ones as it enables machine processing and inter-subjective understanding.

Hierarchical relationships between viewpoints are currently not considered in the MUVIEMOT method. In future version of the tool it is planned to integrate this specific viewpoint relationship along with all necessary improvements and extensions.

Although MUVIEMOT is designed for method engineers, some of the introduced steps might not be intuitive enough. The method therefore needs to be introduced thoroughly, comprising a training and a theoretical introduction to the ideological architecture of the method. The modeler needs to be aware of the specifics of multi-view modeling in order to reflect them appropriately in the conceptual design, e.g., the introduced multi-view modeling principles or the projective and selective relationships between viewpoints and meta models.

Opportunities As pointed to earlier, multi-view modeling is not limited to the domains of enterprise modeling and enterprise architecture. By contrast, multi-viewing is utilized in an ever increasing diversity of domains. Hence, the generic approach can be adopted and utilized to a large scale.

Developing a modeling tool for the MUVIEMOT method and publishing it as open source and open use tool as part of the Open Models Initiative[28] enables the application and evaluation of the tool on a broad basis. With an increasing number of applications, the maturity of the method increases, too.

Threats A major threat to validity of the results of MUVIEMOT evaluation step is the fact, that the persons who developed the conceptual design and the modeling tool are those who analyze its utility. They could use the tool in a way that is not common. Hence, the evaluation results would be tampered (cf. WIERINGA AND MORALI (2012)). As a solution, the tool should be given to independent testers who are familiar with the modeling method.

[28] `http://www.openmodels.at`, last checked: 2015-01-18

The MUVIEMOT method is strongly aligned to the foundations of multi-view modeling and meta modeling. If, in the future, these foundations emerge, the method - or at least parts of it - need to be adopted, too. Moreover, if tool development platforms, e.g., meta modeling environments, further raise the abstraction level and support the tool developer in specifying complex multi-view modeling tools more intuitively and efficiently, MUVIEMOT would become obsolete.

6.5 Summary

In the preceding sections, a modeling method, enabling the development of conceptual designs for multi-view modeling tools has been introduced - the MUVIEMOT method. After the requirements for such a domain-specific modeling method have been identified, the constituents of the MUVIEMOT method have been introduced. The method is aimed at bridging the gap between a multi-view modeling method on the one hand and the development of a conceptual design for a corresponding multi-view modeling tool on the other. Hence, it helps to translate the ontological and theoretical constituents of a modeling method to the conceptual and technical specification of a modeling tool.

Considering the different user roles of MUVIEMOT, a parallel is drawn to the three *"players in the game"* of generating an architecture representation by ZACHMAN (1987). Zachman defines three fundamental players: the *owner*, the *designer*, and the *builder*. *"The owner has in mind a product that will serve some purpose. The architect transcribes this perception of a product into the owner's perspective. Next the architect translates this representation into a physical product, the designer's perspective. The builder then applies the constraints of the laws of nature and available technology to make the product producible, which is the builder's perspective"* (ZACHMAN, 1987, p. 459). Transferring this classification to the MUVIEMOT method and a multi-view modeling method, the steps are most suitably mapped to the user roles as follows:

Method Owner The method owner should create the *Modeling Scenario* step, thereby trying to capture his/her understanding of the multi-view scenario at hand.

Tool Designer The tool designer should then specify the *Modeling Language*, *Modeling Procedure*, and *Viewpoint Dependencies* accordingly. Hence, the designer transforms the informal and intangible understanding of the owner into a more precise and formalized representation.

Tool Developer Finally, the tool developer should realize the conceptual tool design by contrasting the designer's perspective with the specific functionality and constraints of the development platform at hand, comprised in the *Conceptual Design*.

The *Evaluation* step should be performed ideally by all three user roles. The technical realization should be contrasted with the expectations of the method owner, discrepancies should be identified and used to initialize a further iteration of the MUVIEMOT method. It should be noted, that in reality, several user roles might be comprised by one single person, i.e., the method owner is the same person as the tool designer. Validity of the evaluation results considerably depends on the provision of multiple, independent persons. It is therefore important to try to involve independent evaluators.

Table 16 summarizes the MUVIEMOT steps, by describing each step using the criteria: *input*, *objectives*, and *output*.

Table 16: Input, objectives, and output of each MUVIEMOT step

	Input	Objectives	Output
Modeling Scenario	Knowledge about the domain to be modeled, meta modeling, and the multi-view modeling method.	The goal is to define a comprehensive overview of the multi-view setting by identifying the meta models, viewpoints, the relationships between them, and the general way of carrying out multi-view modeling.	A modeling scenario for the multi-view modeling method comprising all viewpoints, the stakeholders and purposes they serve on an informal level.
Modeling Language	Knowledge about the domain to be modeled, meta modeling, and the Modeling Scenario specification.	The goal is to define a formalized specification of the multiple viewpoints' modeling languages by means of meta models. The modeling concepts are derived from meta models.	Meta models and modeling languages of the viewpoints.

Modeling Procedure	Modeling Scenario, Knowledge about the modeling methods' procedure.	The goal is to define all modeling operators the modeling method holds by relating them to the viewpoints they: can be triggered-in; have an effect-on; or have a conditional-effect-on.	A multi-view modeling use case model with a formalized specification of the modeling procedure.
Viewpoint Dependencies	Meta models of the viewpoints, knowledge about the interplay of the multiple viewpoints.	Definition of all dependencies between the viewpoints in a two-step approach: First, syntactic dependencies derived from the viewpoints' meta models are defined. Second, semantic dependencies, not given by overlapping meta models, can be added.	A comprehensive set of all syntactic and semantic dependencies of modeling concepts in all viewpoints.
Conceptual Design	Multi-View Modeling Use Cases, Modeling Dependencies, and optionally knowledge about the tool development platform.	In this phase, all information is composed into one conceptual design (CD) for a multi-view modeling tool. The CD comprises functional requirements, mostly coming from the modeling method, non-functional requirements mostly not method-specific, and optionally some requirements from the development platform.	A conceptual design, comprising the characteristics of a multi-view modeling method. The CD can be platform-independent or mapped to the functionality and possibilities of a certain tool development platform.

Evaluation	Conceptual Design	The goal is to evaluate the created conceptual design e.g., by producing a prototypical implementation and testing it with use cases. In this phase, the CD should be discussed with tool developers.	The results of the evaluation should help to reflect the design decisions taken. Consequently, another iteration of the approach, i.e., some adjustments on the specifications, may be indicated.

This section introduced the MUVIEMOT modeling method. It also covered a first evaluation of the method using the gathered requirements and the identified research questions, and relating them to the constituents of MUVIEMOT. Lastly, a SWOT analysis has been performed, describing strengths, weaknesses, opportunities and threats of the method. In the following, utility of the method is evaluated by means of a comprehensive *illustrative scenario* (PEFFERS ET AL., 2012). Hereby, the MUVIEMOT method will be used to create a conceptual design for the SOM enterprise modeling method.

7 Conceptual Design & Development of a SOM Multi-View Modeling Tool

The MUVIEMOT method is now being evaluated by means of an illustrative scenario (PEFFERS ET AL., 2012). In this regard, the conceptual design of a multi-view modeling tool for the Semantic Object Model (SOM) enterprise modeling method is developed. More precisely, conceptual design for SOM business process modeling and the specification of business application systems is described. For the enterprise plan, no comprehensive modeling method is defined at the time of writing this thesis. Hence, only the modeling language and the dependencies with business process models are presented.

This section is structured as follows: First, a general introduction to the SOM method is given in section 7.1. After focusing on the business process modeling part of the SOM method in section 7.1.2, the model-driven derivation of business application systems from SOM business process models is discussed in section 7.1.3. Section 7.2 then shows the application of MUVIEMOT to SOM by outlining *Modeling Scenario*, *Modeling Language*, *Modeling Procedure*, *Viewpoint Dependencies* and *Conceptual Design* of a multi-view modeling tool for SOM. The section concludes with a brief description of the realized modeling tool[29] in section 7.3. An evaluation of the utility and efficiency is performed in section 7.4 by contrasting the benefits of applying the MUVIEMOT method with the efforts required to gather all requirements for a SOM modeling tool without MUVIEMOT.

7.1 The SOM Enterprise Modeling Method

The Semantic Object Model (SOM) is a comprehensive enterprise modeling method conceptually based on object-orientation and transaction-based coordination. First works on the method have been published in 1990 by FERSTL AND SINZ (1990). Since then, the method has emerged steadily (FERSTL AND SINZ, 1991, 1995, 2006), comprising also additional application scenarios (WAGNER AND FERSTL, 2010; PÜTZ AND SINZ, 2010a; TEUSCH AND SINZ, 2012; HARTMANN ET AL., 2013; WOLF AND BENKER, 2013). The backbone of the SOM method is an enterprise architecture, consisting of three model layers: *enterprise plan*, *business process model*, and *resource model* (see Figure 41).

Enterprise Plan The enterprise plan defines the global task of an enterprise from an perspective outside of the business system (e.g., goals, business strategy, surrounding conditions, resources), thereby modeling with the viewpoints *Object System* and *Target System*.

[29] In the following, referring to the "SOM tool" implies the consideration of the business process modeling and the business application modeling part of the SOM method. Both parts have been considered in the conceptual design whereas the enterprise plan has been omitted due to a missing modeling language specification.

Chances, risks, strengths, and weaknesses faced by the enterprise are also considered in the enterprise plan. This first layer of the SOM enterprise architecture comprises task level and resource level as abstractions.

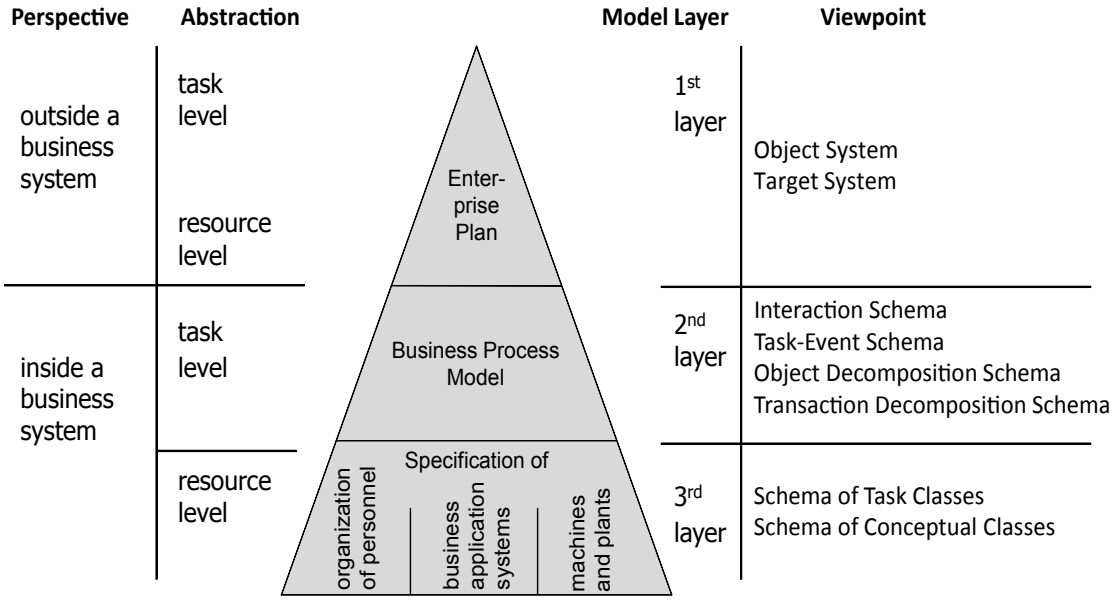


Figure 41: SOM enterprise architecture (extended from (FERSTL AND SINZ, 2008, p. 193))

Business Processes The business process model analyzes the tasks of an enterprise by taking an perspective inside of the business system. The business process model defines a strategy for fulfilling the enterprise plan. The resources needed to execute these tasks are not considered on the second layer of the enterprise architecture. SOM business process models utilize the metaphor of a distributed system, consisting of loosely coupled and autonomous components (FERSTL AND SINZ, 2013, p. 198). Four viewpoints are defined on the second enterprise architecture layer: Interaction Schema (IAS), Task-Event Schema (TES), Object Decomposition Schema (ODS), and Transaction Decomposition Schema (TDS).

Business Application Systems The third layer of the enterprise architecture specifies the resources required for carrying out the tasks of the business processes. Personnel, responsible for the execution of non-automated tasks, is distinguished from business application systems, machines and plants, responsible for the execution of automated and semi-automated tasks. The SOM method concentrates on the specification of the business application system. For their specification, two viewpoints are available: Schema of Conceptual Classes (COS) and Schema of Task Classes (TAS).

Three application scenarios for the SOM enterprise modeling method can be distinguished: (1) creating *to-be* enterprise models, (2) analyzing *as-is* enterprise models, and (3) re-engineering of existing enterprise models (FERSTL AND SINZ, 1996). As these different scenarios share the same modeling procedure and modeling language, the focus in the following is on the several enterprise architecture layers with an emphasis on the multiple viewpoints and the corresponding meta models. On each layer, SOM specifies an integrated meta model. Consequently, the different viewpoints are syntactically integrated. The meta models of each viewpoint are then defined by applying a projection operator onto the integrated meta model.

According to the generic architecture framework (see Figure 27), each layer describes an enterprise as a whole but with respect to a given perspective, i.e., outside on the 1^{st} layer or inside a business system on the 2^{nd} and 3^{rd} layer. Within each layer, viewpoints, dedicated to specific aspects, visualizing a sub-set of the current model layer, are utilized to further decompose the complex model. Each viewpoint is defined by applying a projection operator on the respective integrated meta model. Consequently, Interaction Schema, Task-Event Schema, Object Decomposition Schema, and Transaction Decomposition Schema are all defined as projections onto the business process meta model (see Figure 44), whereas Schema of Task Classes and Schema of Conceptual Classes are defined as a projections onto the meta model for the specification of application systems (see Figure 46).

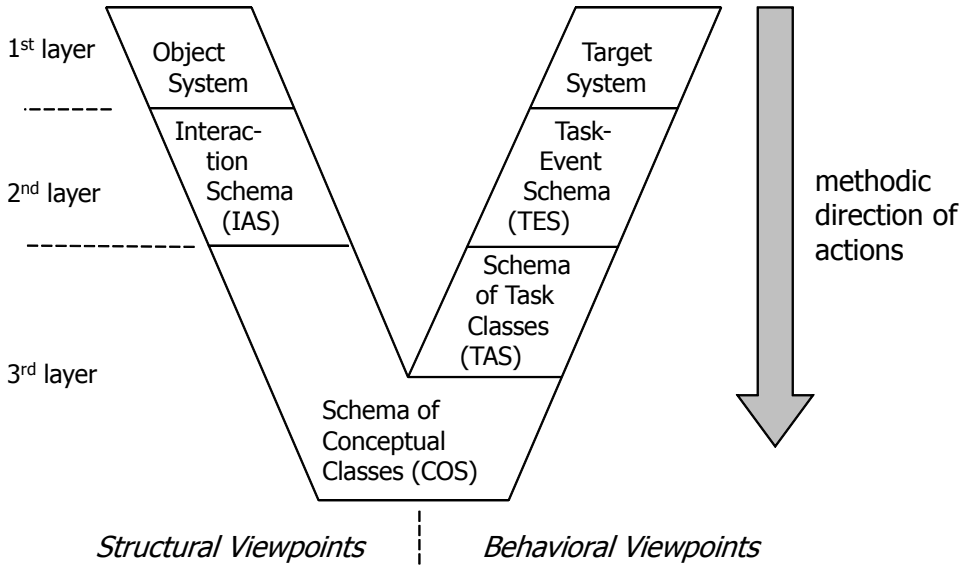


Figure 42: SOM procedure model (cf. (FERSTL AND SINZ, 2008, p. 195))

Aligned to the enterprise architecture (see Figure 41) that structures the method, SOM also defines a *procedure model*. Herewith, the methodic direction of actions suggested to be undertaken in an ideal setting is defined (see Figure 42). SOM utilizes a top-down approach for enterprise modeling. Starting with the first layer, the goals and the strategy for the enterprise

are defined; followed by the business process model specification on the second layer, realizing the strategy for fulfilling these goals; finally on the third layer, business application systems are specified, responsible for carrying out the tasks of the business processes. This methodic direction of actions draws an ideal picture; in reality, depending on the pursued goals of applying SOM, one may start with the business processes. However, following the top-down approach is further motivated by model transformations that are defined between models of the business process layer and the business application systems layer. Hence, a modeler can create initial models of the third layer by transforming models already created on the second layer.

The model transformations are specified on an abstract level by relating meta model elements of adjacent meta models. The integration between the enterprise plan and the business process layer is ongoing research (cf. HARTMANN (2011); HARTMANN AND WOLF (2012)). Therefor, the latest results are briefly discussed in section 7.1.1. The focus is then on the second and third enterprise architecture layer, and the model transformations between these two layers (section 7.1.2 and section 7.1.3). Moreover, the model-driven derivation of Business Process Model and Notation (BPMN) workflow schemata from SOM business process models is discussed in section 7.1.3.3.

7.1.1 Enterprise Plan Modeling in SOM

The first layer of the SOM enterprise architecture aims at specifying the enterprise on a very high abstraction level. By contrast to the second and the third layer, the enterprise plan layer is mostly specified informally. A comprehensive introduction to the enterprise plan layer can be found in (FERSTL AND SINZ, 2013, p. 198f.) and (HARTMANN, 2015).

Metaphor

The construction of an enterprise plan is guided by the metaphor of a *"global enterprise task"* (FERSTL AND SINZ, 2013, p. 198). This global enterprise task is specified by taking an outside perspective on the enterprise. Two facets are distinguished, a behavioral and a structural facet. The former is used to elaborate on distinguishing the system of discourse from its environment as well as the interactions between them by means of service relationships. The latter defines the resources required to fulfill the identified global enterprise task.

Meta Model

All specifications in this layer of the SOM method are on an informal level, i.e., using natural language, supported by conventional business and market analysis methods, e.g., environmental analysis, SWOT analysis, decision models, brainstorming. Hence, there is no comprehensive meta model for the enterprise plan layer of the SOM method. In her phd thesis, HARTMANN (2015) investigated how the enterprise plan can be used as an

instrument for strategically controlling the enterprise. Therefore, a meta model for this purpose has been introduced (see Figure 43) comprising premise, strategy, actions, goals, and measures (cf. (HARTMANN, 2015, p. 107). This meta model enables the codification of the strategic alignment of the enterprise.

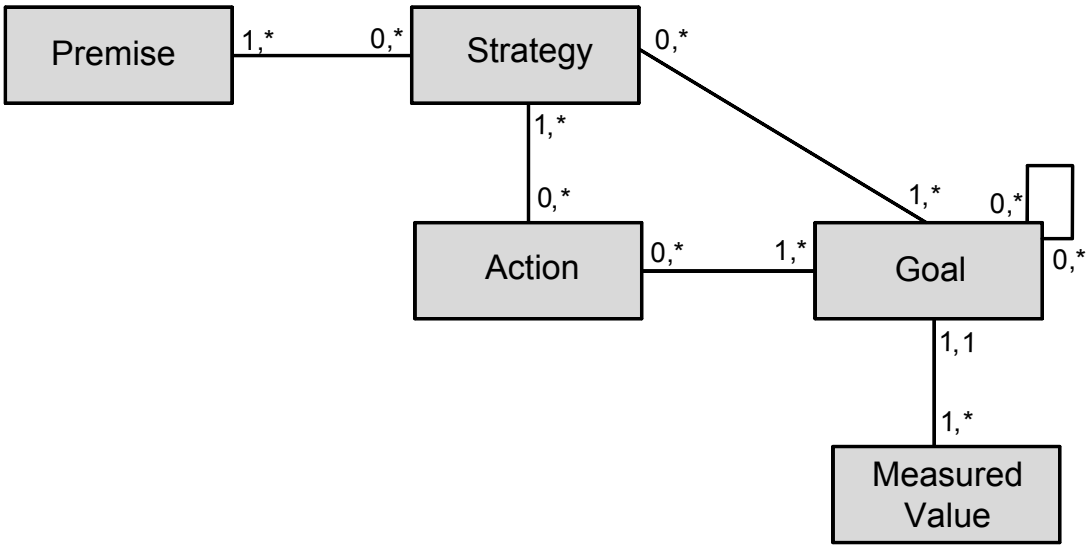


Figure 43: Meta model of the SOM enterprise plan layer (cf. (HARTMANN, 2015, p. 107)

Architecture Model

Initially, the SOM enterprise plan was comprised of two viewpoints, the structural viewpoint, referred to as *Object System*, and the behavioral viewpoint, referred to as *Target System*. The two viewpoints are now introduced briefly:

Object System The Object System defines the global enterprise tasks by distinguishing the relevant system of discourse from its environment. Additionally, services and goods exchanged between the system of discourse and its environment are specified.

Target System The Target System uses the specification of the Object System as an input and generates a strategy for the fulfillment of the identified tasks. It furthermore defines factual goals and formal goals guiding the enterprise's strategic alignment.

Process Model

Initially, due to the lack of a meta model and the informal specification of the enterprise plan, layer two and three of the SOM enterprise architecture have only been coupled on a conceptual basis. Recently, HARTMANN (2015) proposed the alignment of the two adjacent layers by integrating the premises, strategies, actions, and measures with the business process models; and transitively with the resource models. Hence, business objects can be annotated with goals and premises whereas actions can be annotated at tasks of the business process layer.

This annotation enables the alignment of the business processes, and the business application systems indirectly as well, to the global enterprise goal. Moreover, through this annotation, the created business process models and resource models can act as knowledge basis for further processing, e.g., queries and analysis of strategy fulfillment or evaluating the effects of changing the business process behavior on the enterprise strategy.

7.1.2 Multi-View Modeling of Business Processes in SOM

The second layer of the SOM enterprise architecture (see Figure 41) is dedicated to the specification of the enterprise's business processes. SOM business process modeling is grounded on systems theory and the notions of business objects, business transactions, tasks, events, and services (FERSTL AND SINZ, 2006, p. 347). Relating to the constituents of modeling methods according to FERSTL AND SINZ (2008) (see section 2.1.2), the modeling method for business process modeling as part of the SOM modeling method is now introduced. According to BORK AND SINZ (2013), the constituents can be described as follows:

Metaphor

SOM business process modeling 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. Thus, the metaphor combines the basic concept of object-orientation with transaction-based coordination.

Meta Model

The center of the meta-model for SOM business process models is built by the concepts of business object and business transaction. A business object can either be an environmental object or an object of discourse, whereas a business transaction can be an initiating transaction, contracting transaction, enforcing transaction, control transaction, or feedback transaction (this generalization is not illustrated in Figure 44). Two business objects are coordinated by one business transaction. Moreover, a business object can be related with multiple business transactions. A business object comprises one or more business tasks, which are assumed to operate on the same object. Each business transaction is performed by two tasks belonging to different business objects. Tasks belonging to the same object can be connected by an internal event. Moreover, the execution of a task can be triggered by an external event. Each transaction is involved in the coordination or transfer of at least one good or service.

Figure 44 illustrates the meta model of the SOM modeling method for business process modeling. It describes the concepts that can be used in order to create valid business

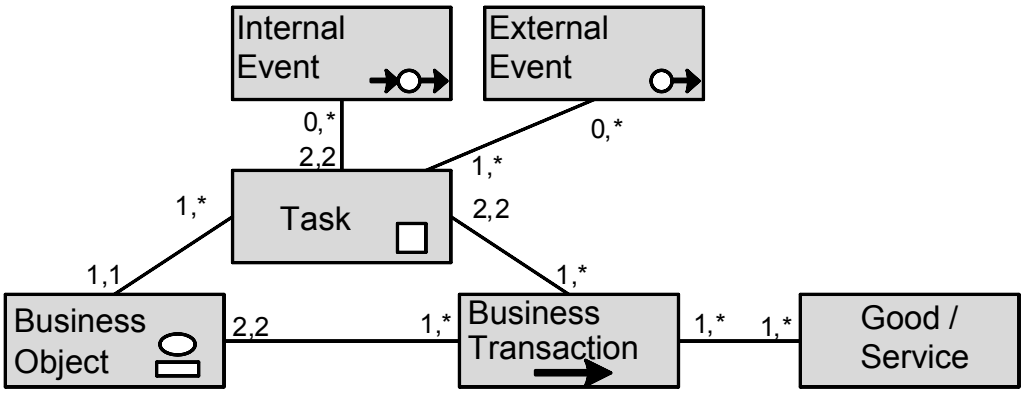


Figure 44: Meta model of the SOM business process modeling method (adopted from (FERSTL AND SINZ, 2008, p. 210)

process models together with cardinalities, constraining the set of valid models. The illustration also shows the notation of the elements by means of a semi-formal specification using shapes. Tasks are visualized by a square, business transactions by an directed arrow, internal and external events by an directed arrow, and in the former case by an circle in the middle of the arrow. Goods and services are not graphically visualized in SOM business process models. The visualization of business objects is twofold: environmental business objects are visualized by an ellipse, business objects of discourse are visualized by a rectangle.

Architecture Model

SOM business process models are represented using a multi-view approach: A structural viewpoint, referred to as *Interaction Schema (IAS)*; a behavioral viewpoint, referred to as *Task-Event Schema (TES)*, a viewpoint on the decomposition of business objects, referred to as *Object Decomposition Schema (ODS)*; and a viewpoint on the decomposition of business transactions, referred to as *Transaction Decomposition Schema (TDS)*. The modeling languages of the viewpoints can be derived as projections onto the SOM business process meta model (see Figure 44). The projection operator, an analogy to the projection operator known in relational databases, delimits the subset of the concepts of the SOM business process meta model to be considered in the viewpoint.

Figure 45 exemplifies the visualization of a multi-view SOM business process model[30]. The process describes an ordering process between a customer and a distributor. The distributor initiates the process by sending the price list to the customer. He or she then files an order and sends it back to the distributor. The process is then concluded by the distributor who sends the ordered products to the customer.

[30] Figure 45 shows a screen shot taken from the SOM multi-view modeling tool realized on the ADOxx meta modeling platform. The tool will be introduced later.

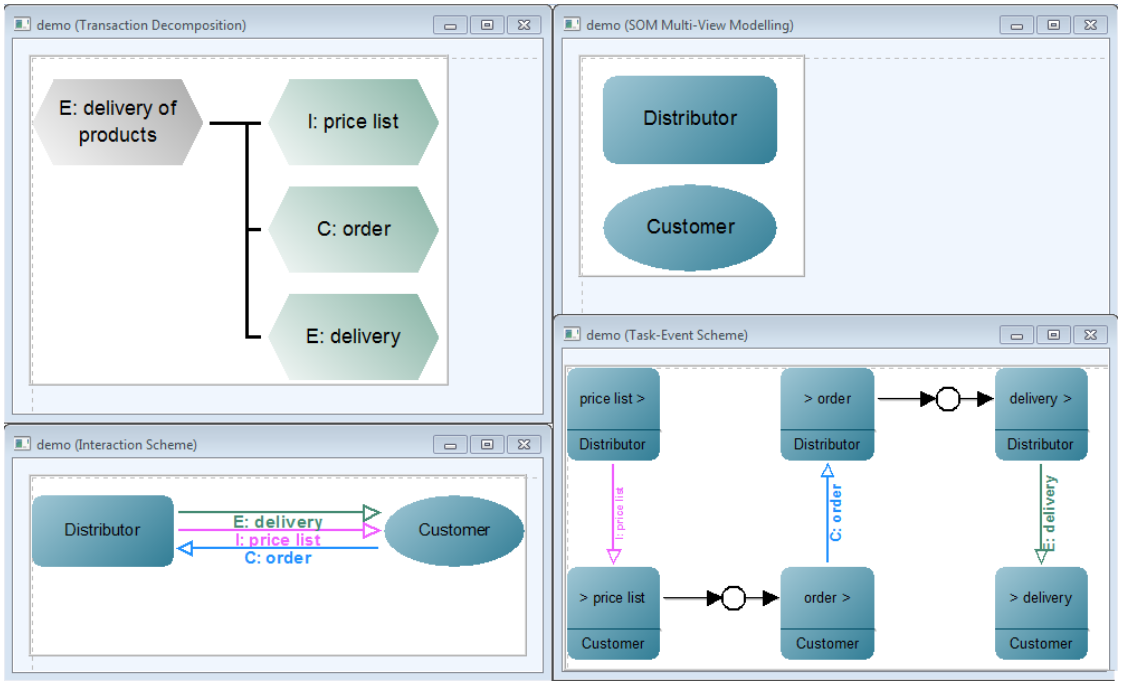


Figure 45: A multi-view SOM business process model

The Transaction Decomposition Schema is visualized on the top left, indicating that an initial enforcing transaction *E: delivery of products* has been decomposed into an initiating transaction *I: price list*, a contracting transaction *C: order*, and an enforcing transaction *E: delivery*. The Object Decomposition Schema is illustrated on the top right side, it shows an environmental object *Customer* and an object of discourse *Distributor*. On the lower left side, the Interaction Schema illustrates the structural viewpoint on the business process. Finally, on the lower right side, the Task-Event Schema comprises a multi-view SOM business process model by a behavioral viewpoint.

The four viewpoints of the SOM business process modeling layer are now introduced briefly:

Interaction Schema The Interaction Schema viewpoint realizes a structural view on the business process. It concentrates on the coordination of business objects by means of business transactions. Consequently, an IAS model is composed of the modeling concepts business object, business transaction, and goods and services.

Task-Event Schema A behavioral view on the business process can be specified with the Task-Event Schema. Using internal events, external events, and business transactions the modeler is able to specify the sequence of tasks a business process execution comprises. The behavioral semantics applied by TES models follows the Petri-net semantics (cf. section 4.2.6).

Object Decomposition Schema The hierarchical decomposition of environmental business objects and business objects of discourse is visualized in a ODS model. Because the decomposition according to the feedback and control principle creates not only decomposed business objects but also business transactions, the ODS meta model is composed of business objects, control and report transactions, and decomposition relationships.

Transaction Decomposition Schema The hierarchical decomposition of business transactions is visualized in a TDS model. This viewpoint comprises initiating transactions, contracting transactions, enforcing transactions, and the decomposition relationships.

The general way of carrying out SOM business process modeling utilizes a multi-view modeling approach. Specifically, SOM business process modeling follows the multi-view by design modeling principle (cf. section 5.4). Hence, all four viewpoints are modified synchronously - changes applied to one view need to be propagated to semantically equivalent changes in the other views. The SOM business process meta model serves as an syntactic and semantic integrator of the viewpoints. Therefore, consistency management needs to be ensured on top of it. Consistency between the four viewpoints is vital for efficiency and utility of the modeling method.

Process Model

The SOM process model defines the concrete steps carried out by a modeler while creating valid SOM models, i.e., one step more detailed in contrast to the architecture model. According to (FERSTL AND SINZ, 2008, p. 205ff.), SOM business process modeling is initiated by separating the system of discourse from the environmental system and identifying the goods and services that are exchanged between them. Hence, the initial SOM business process model is constituted of objects of discourse, environmental objects and enforcing transactions related between them. All of them are on an initial level, i.e., they have not been decomposed yet. The modeler then uses this initial model and refines it according to the system under study. The refinement of SOM business process models is guided by precisely specified **decomposition rules** the modeler must recursively apply to business objects and business transactions. Despite these restrictive modeling technique, SOM also defines **modeling heuristics**, i.e., guidelines and best practices for modelers, developed while utilizing the method in a number of cases. In the following, first the decomposition rules are introduced, then SOM modeling heuristics are briefly described.

SOM utilizes two different decomposition principles that establish the semantics of the **coordination**: According to the **negotiation principle**, non-hierarchical coordination of business transactions between two business objects O and O' can be modeled as sequence

Table 17: Decomposition rules for business objects and business transactions (FERSTL AND SINZ, 2008, p. 203)

Decomposition rules for business objects	
1	$O ::= \{O', O'', T_r(O', O''),\ [T_f(O'', O')]\}$
2	$O ::= \{O', O'', [T(O', O'')]\}$
3	$O ::= \{spec\ O'\}^+$
4	$O', O'' ::= O$
Decomposition rules for business transactions	
5	$T(O, O') ::= [[T_i(O, O')\ seq]\ T_c(O', O)\ seq]\ T_e(O, O')$
6	$T_x ::= T'_x\{seq T''_x\}^+ \mid T'_x\{par\ T''_x\}^+ (for\ x = i, c, e, r, f)$
7	$T_x ::= \{spec\ T'_x\}^+ \qquad (for\ x = i, c, e, r, f)$
8	$T_i \mid T_c \mid T_e ::= T$
9	$T_r \mid T_f ::= T$

::= Replacement	{ } Set	seq sequential relation
[ ] Optional	$\{\}^+$ Repetition (1,*)	par parallel relation
| Alternative	$\{\}^*$ Repetition (0,*)	spec Specialization

of an initiating transaction T_i related from O to O', a contracting transaction T_c related from O' to O, and an enforcing transaction T_e from O to O' (see rule 5 in Table 17). Initiating and contracting transactions are considered optional. They can be omitted e.g., in case of already defined contracts between the coordinated business objects that specify the initiating and contracting steps.

According to the **feedback control principle**, a business object O is decomposed into two business objects: a management object O' and an operational object O'', connected by a control transaction T_r from O' to O'' and an optional feedback transaction T_f in the opposite direction (see rule 1 in Table 17). Management object and operational object are hierarchically coordinated by means of a feedback control loop (FERSTL AND SINZ, 2006, p. 355).

The specialization decomposition enables decomposing a business object or a business transaction into one or more specialized business objects or business transactions of the same type, respectively (see rule 3 and 7 in Table 17). The new created elements are connected with the decomposed element with a decomposition relation. Each business object can be decomposed into two business objects of the same type, connected by any business transaction (see rule 2 in Table 17). Each business transaction can be decomposed into a number of parallel or sequentially scheduled business transactions of the same type (see rule 6 in Table 17). All decomposition rules can be applied recursively (see rule 4, 8 and 9 in Table 17) to produce a precise mapping of the complex system under study.

SOM modeling heuristics have been established as outcomes of multiple instantiations of the SOM method in research, education, and industrial projects. The heuristic knowledge aims at providing guidelines for the decomposition of business objects and business transactions. Furthermore, it aims at helping the modeler in identifying possible decompositions in SOM business process models (FERSTL AND SINZ, 2008, p. 206f.).

- Expose the coordination of business transactions by applying the negotiation decomposition principle (see rule [5] in Table 17).
- Expose the coordination of business objects by applying the feedback control decomposition principle (see rule [1] in Table 17).
- Homogenization of business transactions according to the transferred service or good. A business transaction transferring a compound good or service should be decomposed into sequential or parallel sub business transactions for each part of the good or service (see rule [6] in Table 17).
- Homogenization of business objects according to the generated good or service. A business object generating a compound good or service should be decomposed into sub business objects for each part of the good or service. If a multi-level coordination is revealed thereby, business transactions should be created to relate the respective business objects (see rules [2] and [3] in Table 17).
- Decomposition of business transactions should be performed prior to the decomposition of business objects because their decomposition often reveals the coordination of the affected business objects. Moreover, service processes can be revealed.
- Small feedback control loops. Single-level feedback control loops are preferable as they react faster and are more flexible compared to multi-level ones.
- Complete negotiation protocols. The negotiation between business objects can be analyzed according to completeness by analyzing the negotiation principle.

- No business transaction without a related good or service. Any modeled business transaction must either directly or indirectly participate in the transfer or the coordination of goods or services.
- Separation of task level and resource level (see Figure 41). Business objects not necessarily correspond to the organizational elements of the enterprise.

7.1.3 Multi-View Modeling of Business Appication Systems in SOM

As suggested by the procedure model of SOM (see section 7.1, Figure 42), SOM business process models can be used to derive an initial specification of resources, carrying out the tasks of business processes. Resources comprise personnel, business application systems, and machines and plants (FERSTL AND SINZ, 2013, p. 199). For the specification of resource models, SOM concentrates on the tasks of the business process model that can be carried out automatically or semi-automatically by business application systems.

In the following, the specification framework used in the previous section for the description of the business process modeling layer is applied to the third layer of the SOM enterprise architecture, thereby outlining metaphor, meta model, architecture model, and process model of this layer. Subsequently, the transformations for an initial derivation of a specification for business application systems defined by (FERSTL AND SINZ, 2013, p. 222ff.) are introduced. Section 7.1.3.1 describes the model-driven generation of Schema of Conceptual Classes (COS) models, followed by a description of the model-driven generation of Schema of Task Classes (TAS) in section 7.1.3.2. Lastly, section 7.1.3.3 introduces the derivation of BPMN workflow specifications from SOM business process models. All transformations are based on SOM business process models and a mapping between the respective meta models (i.e., the meta model of SOM business process models, see Figure 44, and the meta model for the specification of applications systems, see Figure 46) referring to the model transformation approach of the Model Driven Engineering (MDE) (OBJECT MANAGEMENT GROUP (OMG), 2003, p. 3-9).

Metaphor

On the third layer of the SOM enterprise architecture, the enterprise system is analyzed from an perspective inside the business system utilizing the metaphor of a socio-technical system consisting of personnel, machines & plants, and business application systems. The SOM resource model concentrates on the information processing tasks, therefore machines & plants are omitted. Moreover, the resource personnel is modeled in conventional organizational models (not discussed here). Hence, the focus of the resource layer is on the model-driven specification of business application systems carrying out automated and semi-automated tasks of the business process models. Semi-automated tasks

are performed by personnel using machines in a partner-partner relationship, therefore benefiting by synergy effects (FERSTL AND SINZ, 2008, p. 194). SOM specifies business application systems as object-oriented and object-integrated distributed systems.

Meta Model

Figure 46 illustrates the meta model for the specification of resource models. *Relationship* is an abstract concept, i.e., it is not instantiated in models; instead, the generalized concepts *is_a*, *interacts_with*, and *is_part_of* are. Figure 46 also visualizes the notation of the elements. Relationships of type *is_a* are visualized by a dotted line, *interacts_with* relationships by a dashed line and *is_part_of* relationships by a solid line. Any relationship connects exactly two *Objecttype* sub types - Objecttypes itself are also abstract. An Objecttype can be of type *object specific*, *service specific*, *transaction specific*, or *task specific*, all visualized by a rounded rectangle. Each Objecttype can be given zero to many *Operators* and zero to many *Attributes*.

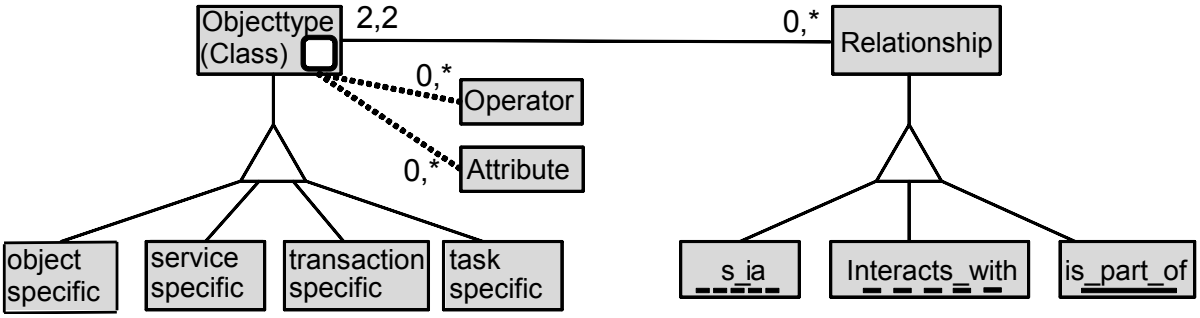


Figure 46: Meta model for the specification of business application systems in SOM (FERSTL AND SINZ, 2008, p. 228)

Architecture Model

The architecture model of the resource level of the SOM method is comprised of two viewpoints on the specification of business application systems: Schema of Conceptual Classes (COS) and Schema of Task Classes (TAS). A COS model is comprised of conceptual objecttypes and their relationships, a TAS is comprised of task specific objecttypes and their relationships, specifying the cooperation of conceptual objecttypes while carrying out business tasks. The modeling language of both viewpoints can be derived as projection onto the SOM business application systems meta model (see Figure 46). Both viewpoints complement each other, therefore they should be consistent to, and do not contradict each other. Two two viewpoints are now described more precisely:

Schema of Conceptual Classes The modeling language of the COS viewpoint is composed of *object specific objecttypes*, *transaction specific objecttypes*, *service specific objecttypes*, and the relationships *is_a*, *interacts_with*, and *is_part_of*. COS

models can be understood conceptually as an extension of the Structured Entity Relationship Model (SERM) (SINZ, 1987, 1988). The quasi-hierarchical graph of SERM diagrams (i.e., existence dependencies are visualized from left to right) is also utilized by COS models. The independent objecttypes (i.e., object specific and service specific) are visualized on the left border of the model, the dependent transaction specific objecttypes are positioned according to their dependency, defined by the business process behavior, from left to right. The cardinalities of the relations in SERM diagrams are also applied for the relationships in COS models. An objecttype in COS models is composed of a *name*, *attributes*, *message definitions* the objecttype can interpret, and *methods* the objecttype provides for handling incoming messages.

Schema of Task Classes The modeling language of the TAS viewpoint is composed of *task specific objecttypes* and the relationships *is_a*, *interacts_with*, and *is_part_of*. With a TAS model, the coordination of objecttypes during the process of carrying out business tasks can be specified. Hence, the TAS always refers to a certain subset of the COS.

Process Model

SOM promotes a top-down approach for creating a comprehensive enterprise model. Therefore, the specification of business application systems using COS and TAS should be built on an existing business process model. The SOM method provides a formalized specification of model transformations between the models of the second and the third SOM layer. Moreover, the method introduces modeling procedures and heuristics for further refinement of the initially derived COS and TAS models.

By contrast to the viewpoints of the business process layer, the viewpoints of the resource layer are not manipulated according to the system-oriented multi-view modeling principle. Changes to the initially derived COS and TAS models are neither reflected in the business process models they have been derived from, nor in the respective counterpart. Hence, SOM utilizes *multi-view by generation* between the second and the third layer.

Figure 47 illustrates the mapping between the meta models for business process models and business application system models. The following sections will describe the mappings and the modeling procedures for refining the initial models in more detail.

7.1.3.1 Model-driven Derivation of Schema of Conceptual Classes

From a methodological point of view, business process models should be transformed into initial business application system models. Figure 47 illustrates the relationships between the

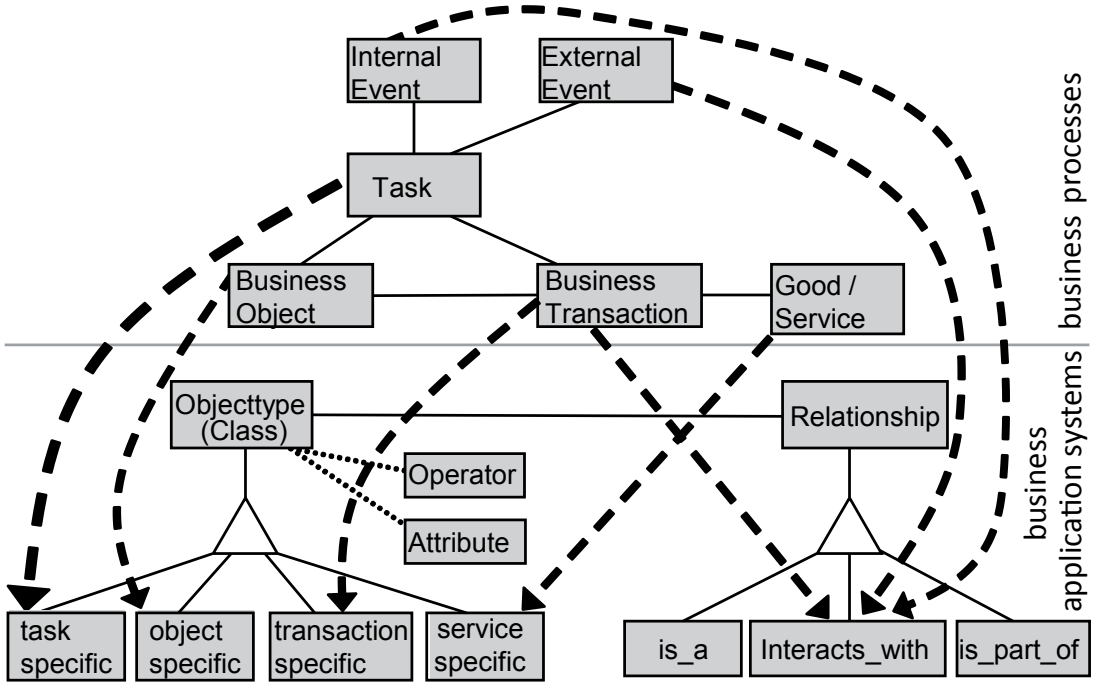


Figure 47: Meta model based transformation of SOM business process models into business application system models

two layers of the SOM enterprise architecture by mapping the concepts of the respective meta models: A *task* is mapped to a *task specific objecttype*; a *business object* to an *object specific objecttype*; a *business transaction* to a *transaction specific objecttype* and an *interacts_with* relationship that connects it directly or indirectly with the corresponding object specific objecttypes and the corresponding service specific objecttypes; *goods and services* are mapped to *service specific objecttypes*; and *internal events* and *external events* are mapped to *interacts_with* relationships that connect the corresponding task specific objecttypes. Notably, *"is_a relationships and is_part_of relationships cannot be linked directly to a business process model. They have to be included during the further specification of the schema of conceptual classes"* (FERSTL AND SINZ, 2006, p. 16).

Meta Model Mapping: Business Process model to COS model

Figure 48 exemplifies the meta model based transformation of SOM business process models into initial Schema of Conceptual Classes models. For readability reasons, the illustration concentrates on the Task-Event Schema as the business process viewpoint with the most valuable information for the COS transformation. However, the transformation requires a comprehensive business process model, e.g., for transforming the business objects and environmental objects of the ODS into corresponding object specific objecttypes. In the meta model mapping on the left side of Figure 48, two transformation rules are numbered. These numbers are used on the

instance level to indicate which mapping rule has been applied (right side of Figure 48).

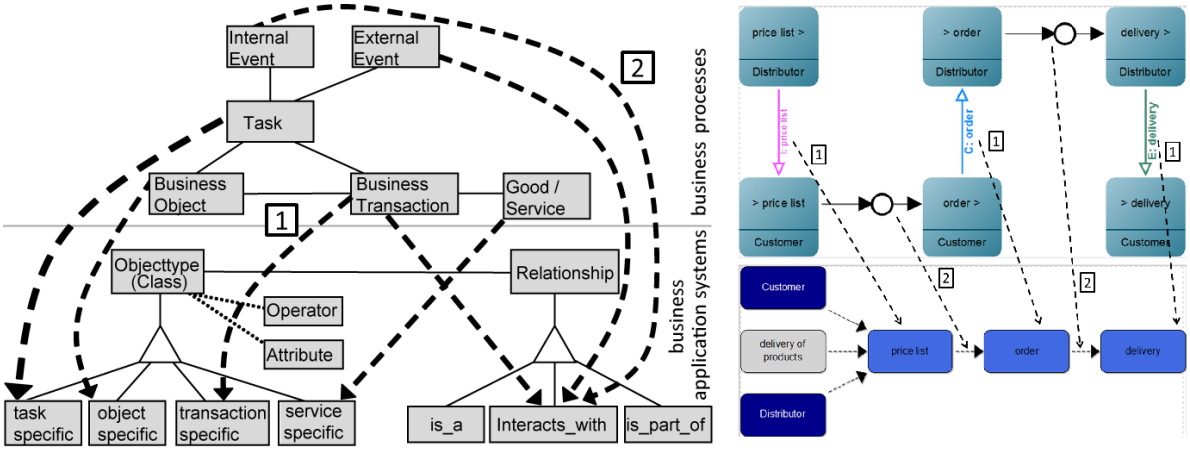


Figure 48: Illustrative transformation of a SOM business process model into a COS model

The initiating transaction *price list*, is transformed into a transaction specific objecttype that is connected with the goal of the business process model *delivery of products* and the corresponding object specific objecttypes *Customer* and *Distributor* by an interacts_with relationship (rule 1 in Figure 48). The same rule has been applied to the contracting transaction *order* and the enforcing transaction *delivery*. However, the created transaction specific objecttypes are only indirectly connected to the object specific and service specific objecttypes. The internal events between task $>$ *price list* and *order* $>$, and between $>$ *order* and *delivery* $>$ have been transformed into interacts_with relationships that are connected to the corresponding task specific objecttypes, respectively (rule 2 in Figure 48).

Schema of Conceptual Classes Modeling Procedure

The initially derived Schema of Conceptual Classes is from a structural perspective quite similar to the Task-Event Schema it is derived from. However, with the transformation the specification of the business application system model is actually starting. (FERSTL AND SINZ, 2013, p. 222f.) defined the following modeling operations for the refinement of initially derived COS models:

- Deletion of all objecttypes if their corresponding business tasks or business transactions cannot be performed automated in reality. Notably, the COS model considers the task layer of an enterprise. It provides a specification of business application systems, hereby only considering the automated tasks.

- Determining the cardinality of the relationships. Depending of the relationship type, the method provides $0, 1$, $0,^*$, $1,^*$ cardinalities for interacts_with and is_part_of relationships,

and $0,1$ and $1,1$ cardinalities for is_a relationships. Depending on the cardinality the notation of the relation changes (FERSTL AND SINZ, 2008, p. 219f.). If the min-value is equals to 0 one line is drawn, if the min-value is equals to 1, a double line is drawn. The max-value effects the end of the relation, i.e., if the max-value is equals to 1 there is only the single or double line, whereas if the max-value is equals to *, an arrowhead is drawn.

- Assigning attributes to the objecttypes. According to the data modeling domain, the method utilizes generalization and normalization relationships between objecttypes. Objecttypes whose corresponding objects or flows are not normalized can be connected by means of a *is_part_of* relationship (e.g., *order position is_part_of order*). Also, objecttypes can be generalized using *is_a* relationships (e.g., *personnel is_a employee* or *is_a contractor*) (FERSTL AND SINZ, 2008, p. 222).

- Defining message formats and operators for each objecttype. If a corresponding TAS exists, relationships in the TAS may indicate additional operators of an objecttype.

- Merging of objecttypes whose message formats and/or operations do widely overlap in order to reduce functional and data redundancy.

7.1.3.2 Model-driven Derivation of Schema of Task Classes

Complementary to the Schema of Conceptual Classes, SOM allows the model-driven derivation of an initial Schema of Task Classes (TAS). A TAS model consists of *task specific objecttypes*, connected by *interacts_with* relationships. Every task specific objecttype describes the collaboration of the objecttypes in the Schema of Conceptual Classes during the execution of a business task. The combination of the task specific objecttypes, by means of the Schema of Task Classes, then describes the workflow of a certain part of the business systems by referring to the relevant part of the business process model.

A task specific objecttype comprises the following constituents: *name* for the objecttype; *attributes* describing the task object of the task fulfilled by the objecttype as a sub-graph of the COS; *message formats* defining the messages to be sent and received by the objecttype; and *operators* defining a solution strategy for the objecttypes' task. In contrast to the COS, the modeler can generate several TAS models for one business process model as each TAS reflects only a part of the business process model, e.g., one TAS model for each business object of the business process model. Hence, the modeler selects the relevant part of the business process model considered in the model transformation.

Meta Model Mapping: Business Process model to TAS model

Figure 49 exemplifies the model-driven transformation of Schema of Task Classes models from SOM business process models. On the left side, the meta model mapping of Figure 47 is illustrated. Three mapping rules are highlighted by numbers: 1) tasks are transformed into task specific objecttypes; 2) business transactions are transformed into interacts_with relationships; and 3) internal events are transformed into interacts_with relationships.

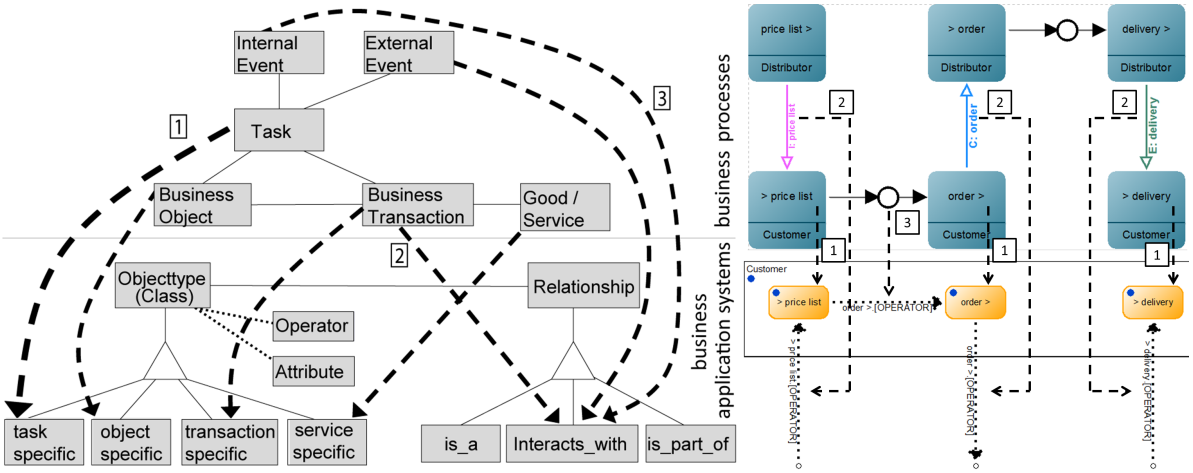


Figure 49: Illustrative transformation of a SOM business process model into a TAS model

On the right side of Figure 49, a concrete Task-Event Schema is visualized on top and the transformed, initial Schema of Task Classes for the business object *Customer* on the bottom. The tasks > *price list*, *order* > and > *delivery* are transformed according to rule 1 into corresponding task specific objecttypes. The initiating transaction *price list*, the contracting transaction *order* and the enforcing transaction *delivery* have been transformed into corresponding in-going and out-going interacts_with relationships, related to the corresponding task specific objecttypes, following rule 2. Rule 3 was then applied to transform the internal event between the tasks > *price list* and *order* > into an interacts_with relationship related to the corresponding task specific objecttypes. The objecttypes are horizontally aligned and clustered in a rectangle according to the business object they belong to.

Schema of Task Classes Modeling Procedure

Similarly to the Schema of Conceptual Classes, the initially derived Schema of Task Classes can be further refined. The refinement is guided by modeling operations, specified by the authors of the approach. In the following, these modeling operations are described briefly: (FERSTL AND SINZ, 2013, p. 233f.):

- Deletion of all task specific objecttypes if their corresponding business tasks cannot be performed automated in reality. As the TAS (like the COS) considers the task layer of an enterprise, it provides a specification of business application systems, hereby only considering automated business tasks.

- Assignment of attributes to each task specific objecttype by means of a sub graph of the corresponding Schema of Conceptual Classes.

- Defining the message formats for each task specific objecttype. The message formats normally correspond to the semantics of the connected interacts_with relationships.

- Specification of the operations by means of a navigation on the sub graph of the Schema of Conceptual Classes assigned to the attributes.

- Merging of objecttypes whose corresponding tasks require, for semantic integrity reasons, to be executed together, e.g., submitting an order and acknowledging an order.

7.1.3.3 Model-driven Derivation of BPMN Workflow Schemata

BPMN is a graphical modeling language developed and standardized by the OMG for business processes modeling and the execution of those models in workflow engines (OBJECT MANAGEMENT GROUP (OMG), 2011a). Version 1.1 of the BPMN standard has been released in 2008 (OBJECT MANAGEMENT GROUP (OMG), 2008), since then the standard has been refined and extended. In the current version, BPMN version 2.0 (OBJECT MANAGEMENT GROUP (OMG), 2011a) released in 2011, a large set of elements is provided to the modeler in order to create precise and detailed business process models. Especially in the latest releases, emphasis has been put on the execution semantics of the business process models in workflow engines, e.g., using Business Process Execution Language (BPEL) or XML Process Definition Language (XPDL) as specification languages (OUYANG ET AL., 2006; JUNG ET AL., 2004; WHITE, STEPHEN A., 2005). Several elements that have been introduced accordingly are motivated be their execution semantics and not solely on their capability of capturing real live phenomenon of business processes more adequately, e.g., the event-driven instantiation gateways.

The large number of BPMN elements leads to discussions between researchers and practitioners, e.g., (BÖRGER, 2012; KOSSAK ET AL., 2012). *"Often the BPMN standard specifies an element in a very general way in one place, and then constrains this description in various other places. After putting all the different descriptions of one element together, we have sometimes found apparent inconsistencies or even contradictions, while at the same time, the semantics of certain elements remains ambiguous"* (KOSSAK ET AL., 2012, p. 53).

Besides these discussions, mostly based on the inexact semantics for the numerous elements in the latest versions, a stable subset of the most commonly used elements of the BPMN standard has emerged. Modeling tools, like the ADOxx Community Edition[31], provide several modes of BPMN modeling, i.e., a standard mode and an expert or comprehensive mode. In the former case, only the commonly used subset of the BPMN elements is usable, whereas the latter supports modeling according to the complete BPMN standard. Albeit all these negativism, a wide acceptance and usage of BPMN in the business process and workflow modeling communities can be stated.

PÜTZ AND SINZ (2010a,b) report on the transformation of SOM business process models into BPMN workflow specifications. The motivation of the approach is to overcome the semantic gap between SOM business process models and BPMN workflow models. The authors argue that the relation of business processes to goods and services is not given in the BPMN. However, the BPMN is capable of specifying the execution of business processes on a lower abstraction level.

Meta Model Mapping: Business Process model to BPMN model

Following the argumentation of the former section, the transformation of SOM business process models into BPMN workflow models is based on the commonly used subset of the BPMN elements. This section describes the mapping in more detail by means of relating the concepts of the SOM business process meta model (see see Figure 44) to the meta model of the relevant subset of the BPMN meta model. Figure 50 illustrates the meta model mapping in a compact way. The conception of the mapping itself is motivated and described comprehensively in PÜTZ AND SINZ (2010a,b).

The transformation between a SOM business process model and a BPMN workflow schemata starts with a sufficiently refined Task-Event Schema (TES), the behavioral view on a SOM business process model. The result of the transformation is an initially derived, valid BPMN workflow model. The mapping between the constituents of the two meta models is now described in more detail: business objects, i.e., objects of discourse and environmental objects, are transformed into BPMN pools; business transactions, related between two business objects are transformed into BPMN message flows, related between the corresponding BPMN pools, realizing the workflow choreography. Internal events of the TES, connecting two tasks of the same corresponding business object are transformed into BPMN sequence flows, related between the corresponding BPMN activities within one BPMN pool. If pre- and post-conditions, e.g., *and, or, xor*, have been defined for a task, the corresponding internal event is transformed additionally to a BPMN gateway. Every task of the Task-Event Schema is mapped to a BPMN

[31] Adonis Community Edition homepage: `http://www.adonis-community.com/`, last checked: 2015-04-13

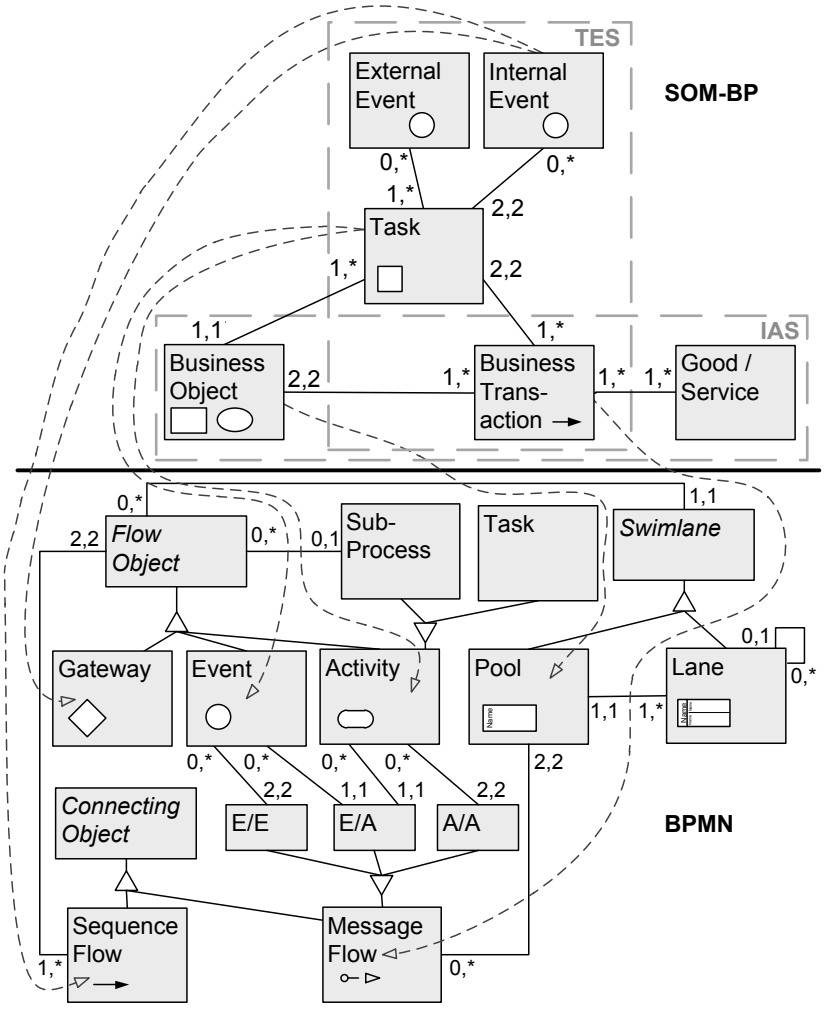


Figure 50: Meta model based transformation of SOM business process models to BPMN workflow schemata (PÜTZ AND SINZ, 2010a)

activity and to an BPMN event where appropriate i.e., due to the different execution semantics of the TES model (Petri-Net) and the BPMN (algorithmic) (PÜTZ AND SINZ, 2010a, p. 62).

Figure 51 exemplifies the meta model based model transformation by means of an illustrates example. The illustrated BPMN model is based on the same SOM business process model as the former two transformations (cf. Figure 45). For the two business objects involved in the business process, i.e., the *Customer* and the *Distributor*, BPMN pools have been created. The tasks of the Task-Event Schema have been transformed into corresponding BPMN activities. The three business transactions have been transformed into BPMN message flows, and the internal events into sequence flows, related between the corresponding BPMN activities. Notably, the example only shows the application of selected mapping rules. The interested reader is referred to PÜTZ AND SINZ (2010a) for comprehensive description of the mapping comprising a more detailed example.

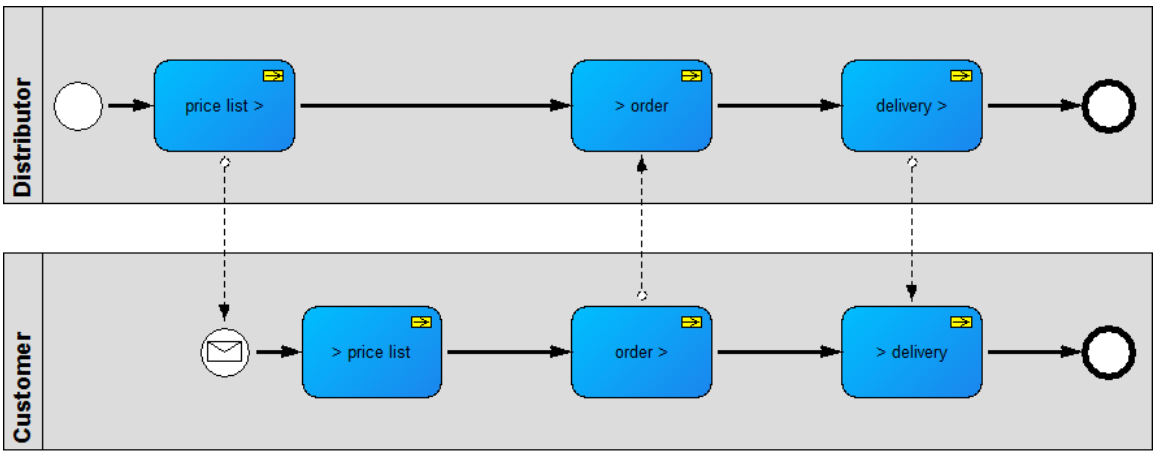


Figure 51: Illustrative transformation of a SOM business process model into a BPMN model

Notably, every pool has its own start event and end event. The *Distributor* pool has the global start event as it is the first task in the Task-Event Schema. Hence, in this case, one task has been transformed into a *start event* connected with a *sequence flow* to an *activity*. The start event of the *Customer* pool is a *message event*, initiated by the message flow from the *price list* > task. Finally, the last task of the Task-Event Schema, > *delivery* concludes the process. Therefore, this task has been transformed into a corresponding *activity*, related with an *end event* by a *sequence flow* relationship. Additionally, every last activity in a BPMN pool is connected with an end event by a sequence flow relationship.

BPMN Modeling Procedure

The initially derived BPMN model can then be further refined, e.g., by merging activities of one pool into sub processes; by moving activities into a new pool; by adding or removing activities that are not included in the business process model or not automatable in the workflow, respectively; or by editing the derived events used in the message flows, i.e., changing the type or the flow dimension. As the modeling procedure is not specific for the SOM method, the interested reader is referred to the BPMN standard (OBJECT MANAGEMENT GROUP (OMG), 2011a) for a comprehensive description.

7.2 Conceptual Design of a SOM Multi-View Modeling Tool

After the introduction to the Semantic Object Model (SOM) modeling method in the precedent section, the following section describes the specification of a conceptual design of a SOM modeling tool. The conceptual design is developed by applying the MUVIEMOT modeling method (see section 6) to SOM. The case study serves as an illustrative scenario (PEFFERS ET AL.,

2012), hereby evaluating the utility of the MUVIEMOT method.

The following sections are structured accordingly to the different steps of the MUVIEMOT lifecycle (cf. section 6.3 and Figure 33). The case study, like the SOM method in its current and mature version, concentrates on the business process and resource layer of SOM. The enterprise plan layer is not specified thoroughly yet, therefore, only a modeling scenario has been generated. The other MUVIEMOT steps omit this layer of the SOM method and concentrate on the other two.

7.2.1 Step I: Modeling Scenario

In the first step of the MUVIEMOT method, a modeling scenario is to be specified; depicting the situation of a human modeler while perceiving the real world and creating a multi-view model representation of his/her perception. In case of SOM, a modeling scenario has to be defined for each of the three layers of the enterprise plan. Each layer has a distinct meta model, a distinct metaphor, and multiple viewpoints. In the following, the three modeling scenarios are described briefly following a top down approach accordingly to the methodic direction of actions, suggested by the SOM enterprise architecture (cf. Figure 41).

7.2.1.1 Enterprise Plan Layer Modeling Scenario

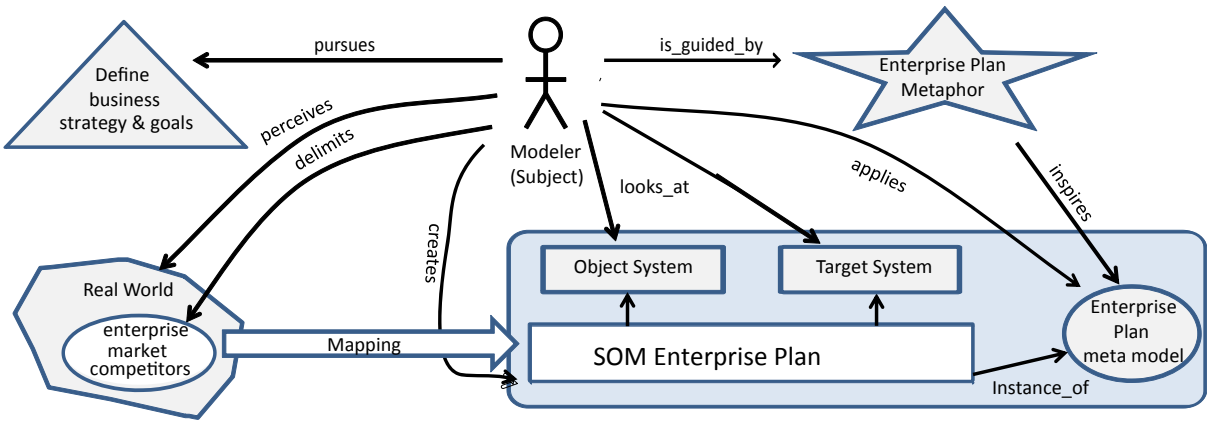


Figure 52: Modeling Scenario for the SOM enterprise plan layer

The modeling scenario of the first layer, the **enterprise plan**, is schematically depicted in Figure 52. The modeling language applied in this layer of the SOM method is not as formally specified as for the other two layers (HARTMANN, 2011, p. 13). In this layer, several potentially informal methods, e.g., mind maps or SWOT analysis can be applied in order to analyze the position of the enterprise in the market, to determine the enterprise strategy, to define the global goals, and to evaluate risks. HARTMANN (2011) started to investigate on the specification of

a meta model for the behavioral part of the enterprise plan. Central modeling concepts of this meta model are actions, strategies, strategic goals, and measured values. However, the work is considered ongoing research. At the time this thesis has been concluded, the research has not been finalized; work on the alignment between the enterprise plan model and the business process model was still ongoing.

The metaphor of the enterprise plan layer is a global enterprise task. The enterprise plan is used to specify this global enterprise task according to its objectives and goals by taking an perspective outside of the business system. Relationships between objects of discourse and environmental objects, i.e., the *object system viewpoint*, are comprised with a global strategy and resources needed to fulfill the enterprise tasks, the *target system viewpoint*.

7.2.1.2 Business Process Layer Modeling Scenario

The goal of the business process modeling layer is to investigate the business processes of an enterprise from an perspective inside the business system. The metaphor is 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 (see section 7.1.2). SOM business process models are described by utilizing a multi-view approach. Four viewpoints haven been defined as projections onto the business process meta model: *Interaction Schema*, *Task-Event Schema*, *Object Decomposition Schema*, and *Transaction Decomposition Schema*. Figure 53 schematically illustrates the modeling scenario for SOM business process modeling.

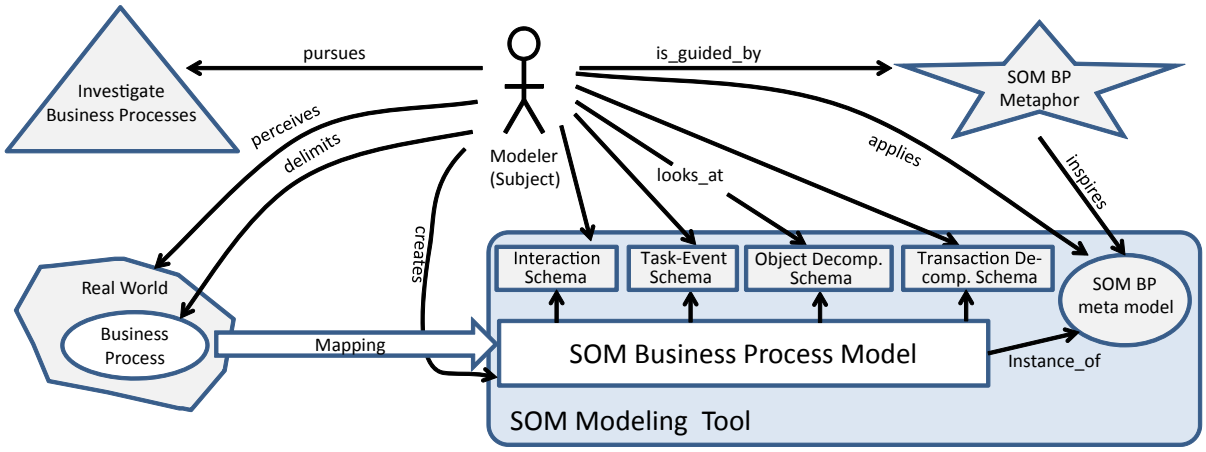


Figure 53: Modeling Scenario for the SOM business process modeling layer (BORK AND SINZ, 2013)

7.2.1.3 Resource Layer Modeling Scenario

On the third layer of the SOM enterprise architecture, resources responsible for carrying out the tasks defined in the business process layer are specified. The metaphor of a socio-technical system consisting of personnel, machines & plants, and business application systems is guiding the modeler in perceiving the real world and delimiting the relevant aspects. Again, a multi-view approach is utilized for the specification of business application systems. Two viewpoints, *Schema of Conceptual Classes* and *Schema of Task Classes* comprise a business application system specification. The modeling languages of both viewpoints can be derived by a projection onto the business application system meta model. Figure 54 schematically illustrates the modeling scenario for SOM business application system modeling on the resource layer.

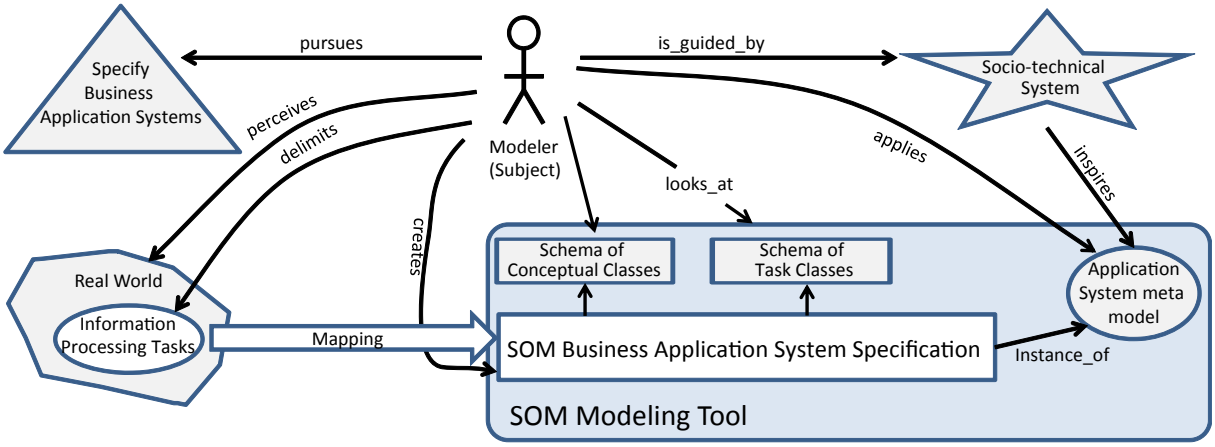


Figure 54: Modeling Scenario for the SOM resource layer

7.2.2 Step II: Modeling Language

The second step of MUVIEMOT focuses on the specification of the modeling language by considering three aspects: 1) the meta models, 2) the viewpoints, and 3) viewpoint-specific attributes (cf. section 6.3.2). The business process meta model of the SOM business process layer has already been introduced (see Figure 44). Also, section 7.1.2 described the constituents and the semantics of the multiple viewpoints on a SOM business process model. Considering the third aspect of the modeling language step, i.e., viewpoint-specific attributes, additional information has been specified.

Depending on the viewpoint, SOM utilizes different notations for business transactions. Within the decomposition schemata, business transactions are visualized as rounded rectangles in a tree-based hierarchy of decompositions. In the Interaction Schema and the Task-Event Schema, however, business transactions are visualized by means of relations, i.e., as directed arrows indication the transaction from outgoing business objects to ingoing business objects, or

sending tasks to receiving tasks in the Task-Event Schema, respectively.

The meta model of the resource layer (see Figure 46) and the modeling language and semantics of the two resource layer viewpoints have been specified in section 7.1.3. Viewpoint-specific attributes are realized for the transformation of business transactions. If transformed into a Schema of Task-Classes, business transactions are visualized by means of a relation connected by the ingoing or outgoing task specific objecttype. By contrast, if transformed into a Schema of Conceptual Classes, business transactions are visualized by means of a rectangle connected with the corresponding object specific, transaction specific, or service specific objecttypes by means of interacts_with relationship.

7.2.3 Step III: Modeling Procedure

In the following, the modeling procedure of SOM enterprise modeling is defined by means of *Multi-View Modeling Use Cases*. Following the specification of these use cases in section 6.3.3, each use case specification is comprised of relationships to other use cases by means of *include*, *extend*, or *generalization*; and relationships to viewpoints by means of *triggered-in*, *effect*, or *conditional effect*. For readability reasons, the specification is visualized in a tabular notation in Table 18. The viewpoints are indicated by their abbreviation: Interaction Schema (IAS), Task-Event Schema (TES), Transaction Decomposition Schema (TDS), and Object Decomposition Schema (ODS). If a certain use case can be triggered in a certain viewpoint, the corresponding cell is marked with an 'T'. Effect relationships are visualized by an 'E', conditional effect relationships are visualized by a 'C' in the corresponding cell.

Table 18: Multi-View Modeling Use Cases of SOM business process modeling (cf. (BORK AND SINZ, 2013))

Use Case	Triggered In				Effect On				References
	IAS	TES	TDS	ODS	IAS	TES	TDS	ODS	
1. Decompose Transaction	T	T	T		C	C	E		
2. Decompose Object	T			T	C	C		E	
3. Revoke Decomposition	T	T			C	C	C	C	
4. Zooming			T	T	E	E			

5. Add Environmental Object				T	E	E		E	Include(6)
6. Add Enforcing Transaction			T		E	E	E		
7. Remove Environmental Object				T	E	E		E	Include(8)
8. Remove Enforcing Transaction			T		E	E	E		
9. Increase Business Process Level	T	T			E	E			
10. Decrease Business Process Level	T	T			E	E			
11. Switch Transaction Direction	T	T			E	E			
12. Define Process Behavior		T				E			
13. Delete Internal Event		T				E			
14. Add External Event		T				E			

Legend: T $\widehat{=}$ Triggered-In, C $\widehat{=}$ Conditional Effect, E $\widehat{=}$ Effect

The table visualizes only the business process modeling use cases. Two use cases of Table 18 are now described in order to illustrate the procedure and the benefit of the multi-view modeling use cases. Generally, the precise specification of the use cases and the conditional effect relationships is performed in the conceptual design step of MUVIEMOT (cf. section 7.2.5 for the Conceptual Design of the SOM tool).

Use case *4. Zooming* depicts the functionality of zooming-in and zooming-out of defined SOM business process levels. A business process level is determined by a certain set of currently visualized business objects and business transactions. SOM defines a zoom operator by means of immediately switching between different, already defined, business process levels, i.e., in order to foster understanding and evaluate the correctness of the model. More precisely, a certain hierarchy level of the Object Decomposition Schema and the Transaction Decomposition Schema is referred to as a business process level. Hence, the zooming use case can be triggered in the ODS and TDS viewpoints. Changing the visualized business process levels necessarily effects to adopt the Interaction Schema and the Task-Event Schema to the selected

level.

Use case *5. Add Environmental Object* is provided to adapt the initial SOM multi-view business process model, consisting of one environmental object, one object of discourse and one enforcing transaction. With this use case, the modeler is able to add more environmental objects, e.g., customer and vendor, in order to match the business process to the system under study. However, as the business process meta model (cf. Figure 44) indicates, a business object always has to have at least one related business transaction. Therefore, this use case is referencing use case *6. Add Enforcing Transaction* by an include relationship. Hence, environmental objects can only be added, if at least one additional business transaction is added simultaneously.

7.2.4 Step IV: Viewpoint Dependencies

In the *Viewpoint Dependencies* step of the MUVIEMOT approach the focus is on the dependencies between the multiple viewpoints. Some of these dependencies have a syntactical origin, i.e., they result from overlapping concepts of the viewpoints' meta models, others have a semantical origin, i.e., they result from the semantically overlapping areas of the viewpoints (cf. section 2.2.3).

7.2.4.1 SOM Business Process Modeling Layer Viewpoint Dependencies

In the business process layer of the SOM method, several viewpoint dependencies have been identified. First, syntactic dependencies are summarized in Table 19. The concepts, specified in the SOM business process modeling meta model (cf. Figure 44) are positioned in the first column. The second column then lists all viewpoints a certain modeling concept is considered in.

Table 19: Syntactic viewpoint dependencies in SOM business process modeling

Modeling Concept	Viewpoints
Business Object	Interaction Schema, Object Decomposition Schema
Business Transaction	Interaction Schema, Task-Event Schema, Transaction Decomposition Schema
Task	Task-Event Schema
Internal Event	Task-Event Schema
External Event	Task-Event Schema
Good / Service	— (not reflected in a distinct modeling concept)

It can be derived, that concepts like business transaction and business object are considered

in multiple viewpoints. Also, some concepts like task and internal/external events are only considered in one viewpoint. Hence, the diversity of syntactic viewpoint dependencies is very high.

The identified syntactic viewpoint dependencies need to be comprised by investigating the semantic overlaps between the viewpoints in order to identify semantic viewpoint dependencies. Those dependencies are not easy to identify. One needs to have a deep understanding of the modeling method at hand in order to identify these dependencies. Most of the semantic dependencies are not one-to-one copies of the same concept in multiple viewpoints (by contrast to the syntactic viewpoint dependencies). The semantic dependencies are often reflected in attribute values of different modeling concepts that need to be kept consistent. Table 20 illustrates the semantic viewpoint dependencies of the SOM business process modeling layer.

Table 20: Semantic viewpoint dependencies in SOM business process modeling

Modeling Concept:*Attribute*	Modeling Concept:*Attribute*
Business Object:*Name*	Task:*Referenced Object*
Business Transaction:*Name*	Task:*Connected Transaction*
Business Transaction:*	Business Transaction:*

As indicated in the first row of Table 20, the value of attribute *Name* of a business object has a semantic dependency to the attribute *Referenced Object* of the modeling concept task. Hence, changing the name of a business object needs to be reflected by accordingly performed changes of the task and vice versa. The second row specifies a dependency between the attribute *Name* of business transactions and the attribute *Connected Transaction* of tasks. In the last row, the * is used to illustrate, that all attributes between two different modeling concepts are dependent. Business transactions in the SOM method are used twofold: First, in the Transaction Decomposition Schema as modeling classes; second, in the Interaction Schema and the Task-Event Schema as relation classes. However, both are visualizations of one and the same aspect of the system under study. Hence, all their attribute values need to be kept consistent.

7.2.4.2 SOM Resource Layer Viewpoint Dependencies

In the resource layer of the SOM method, no viewpoint dependencies are given. Although the Schema of Task Classes and the Schema of Conceptual Classes share a semantic overlap, the SOM method does not specify dependencies. Both viewpoints are generated from the business process viewpoints as a snapshot, consistent at the time created. No changes performed to this initially derived models, are propagated to the business process models. Moreover, changes in one of the resource layer models are not reflected in the other. Hence, whereas the multiple

views of the business process layer are tightly coupled and simultaneously modified (following the multi-view by design principle), the views of the resource layer are only loosely coupled through the common business process model they are derived from (following the multi-view by generation principle).

7.2.5 Step V: Conceptual Design

The conceptual design is comprised of the information of the preceding MUVIEMOT steps. The collected information is enhanced and precised in order to specify the functionality, i.e., the functional requirements of a multi-view modeling tool with an emphasis on view-consistency. Furthermore, this specification also covers non-functional requirements targeting at efficiency and utility of the tool. Table 21 summarizes the functional requirements of the business process modeling layer of the SOM method by referring to the Multi-View Modeling Use Cases of section 7.2.3. If applicable, the tool functionality is structured by means of relating modeling actions (i.e., *do-operator*) with their corresponding recovery operation (i.e., *undo-operator*).

Table 21: Functional requirements of a SOM multi-view modeling tool

Function	Do-Operator	Undo-Operator
Decomposition	Decompose	Revoke Decomposition
Navigation	Zoom-In	Zoom-Out
Edit business process level	Increase business process level	Decrease business process level
Edit initial business process model	Add enforcing transaction	Remove enforcing transaction
	Add environmental object	Remove environmental object
Process behavior	Add internal event	Remove internal event
	Add external event	Remove external event
Transaction direction	Switch transaction direction	
Model Transformation	Generate COS Generate TAS Generate BPMN	

In the Conceptual Design step, each requirement is described by the criteria: *Function*, *Object*, *Operator*, *Effect*, and *Consistency* (cf. section 6.3.5). In Table 22, the conceptual design

specification for several multi-view modeling use cases is exemplified.

The first functional requirement is the *Decomposition*. The decomposition function however, can be applied to *Business Objects*, defined as **Decompose Object**, and *Business Transactions*, referred to as **Decompose Transaction**. The effect of decomposing a business object is as follows: After the modeler has triggered the operator on a specific business object, he or she has to select one of the applicable SOM decomposition rules (see Table 17). In case of a business object decomposition, *hierarchical, non-hierarchical, and specialization decomposition rules* can be applied; in case of business transaction decomposition, *negotiation, specialization, sequencing, and parallelization* can be applied.

The tool then generates the decomposition products and connects them with the decomposed object by means of a decomposition relationship. The modeler then specifies the semantics of the performed decomposition by editing the attribute values (e.g., of the attribute name). As the SOM method specifies a set of different decomposition rules, the effects of applying them vary significantly. In case of a business transaction decomposition, independently from the applied rule, no immediate changes to other views are necessary. However, decomposing business objects might affect other views than the Object Decomposition Schema depending on the applied rule. If the modeler decides to specialize a business object into two business objects, he or she is able to add business transactions that are routed between the decomposition products. If non-hierarchical decomposition is applied to business objects, the modeler even must create a new enforcing transaction between the decomposition products. Consequently, these decompositions also affect the Transaction Decomposition Schema. The applied decomposition rules must be considered in the consistency management to ensure a consistent state of the model.

Besides the decomposition functionality, SOM also provides the corresponding undo-operator (cf. Table 21), i.e., the *Revoke Decomposition* operator. This operator can be applied to business objects and business transactions in the respective decomposition trees. Whenever the modeler triggers the revoke operator, the child nodes of the selected node in the corresponding decomposition trees are removed from the entire SOM business process model. Hence, they are removed from all views currently visualizing these business objects or business transactions, respectively. Similarly to the decomposition, the revoke operator has to be considered in the consistency management, as the effects of revoking a decomposition in the decomposition trees might affect multiple views, depending on the formerly applied decomposition and the decomposition products that need to be deleted.

Table 22: Conceptual Design specification for a SOM modeling tool

Function	Decomposition
Object	Business Object
Operator	*Decompose Object*
Effect	The selected business object is decomposed according to a certain SOM decomposition rule. Depending on the chosen rule, new business transactions must be created, e.g., if the negotiation principle decomposition is applied. Object Decomposition Schema and, depending on the applied decomposition rule, the Transaction Decomposition Schema must be updated.
Consistency	If the negotiation decomposition principle for object decomposition is applied, the modeler must create one or more additional business transactions which must be included and kept consistent in the Transaction Decomposition Schema. The decomposed object in the Interaction Schema and the corresponding tasks in the Task-Event Schema should visualize the performed decomposition, i.e., the existence of a higher decomposition level.
Function	Decomposition
Object	Business Transaction
Operator	*Decompose Transaction*
Effect	The selected business transaction is decomposed according to a certain SOM decomposition rule. All new created business transactions must be visible in the Transaction Decomposition Schema. The decomposed transaction, and all references to this transaction in the Interaction Schema and Task-Event Schema must be updated in order to visualize the performed decomposition to the modeler.
Consistency	Depending on the selected decomposition rule, one or more business transactions of the same or of a different type must be created and visualized in the Transaction Decomposition Schema as child nodes of the decomposed transaction.

Function	Decomposition
Object	Business Object
Operator	*Revoke Decomposition*
Effect	The child nodes of the selected object in the Object Decomposition Schema are deleted from the SOM business process model. If the selected object has been decomposed according to the feedback control principle, the created feedback and control transactions are to be removed, too. If enforcing transactions were related between the child nodes, these transactions need to be removed as well.
Consistency	If the modeler created additional transactions during the formerly performed decomposition of the selected objects, these transactions have to be deleted, too. Moreover, all stored information about transactions, connected with the objects to be deleted, must be deleted, too.

Function	Zooming
Object	Business Object, Business Transaction
Operator	*Zoom on selected Level*
Effect	The currently in Interaction Schema and Task-Event Schema visualized business process level is updated to an already defined more or less detailed decomposition level. The visualized level depends on the business object and business transaction and its position in the corresponding decomposition trees. Executing the zoom operator does not specify any new information to the SOM model. By contrast, zooming is considered only a visualization operator. Hence, it can only be applied to business process levels that have been defined formerly by the modeler.
Consistency	The relationships between business objects and business transactions within all views must be consistent. The visualized process model must be the same as defined formerly in the modeling process, i.e., correct preservation of business process levels is a prerequisite of applying the zooming operator.

Function	Add Model Element
Object	Business Process Model

Operator	*Add Environmental Object*
Effect	An additional environmental object is created and added to the SOM model. It must be connected with a new enforcing transaction to an already existing business object in order to comply with the cardinality rules, specified in the SOM meta model.
Consistency	The new elements, i.e., the business object and the new business transaction, must be visualized in all four SOM views. The relationship between the new business object and the existing one must be consistently visualized in the Interaction Schema and Task-Event Schema.

Besides functional requirements, non-functional requirements comprise a overarching Conceptual Design. Table 23 therefore describes a set of non-functional requirements that are specific for SOM modeling. However, conventional NFRs like stability, usability and so forth should be also considered. However, these NFRs are neither specific for SOM nor for multi-view modeling tools and therefore omitted in the following.

Table 23: Non-functional requirements of a SOM multi-view modeling tool

Category	Operator	Description
Modeling	Separate decomposition and refinement	The modeler should be able to decompose several times, before he or she refines the business process model to a more or less detailed level, i.e., increase or decrease level of business process.
	Zooming	The modeler should be able to immediately switch between already defined business process levels without deleting the currently visualized business process level. This enables to obtain an overview if the models get bigger.

Visualization	Smooth Edges	This layout algorithm is responsible for changing the visualization of all business transactions in the Interaction Schema. All relations should be routed in a direct way between business objects.
	Auto-Layout	This layout algorithm is responsible for changing the visualization of all business transactions in the Interaction Schema. All relations should be routed either directly or by adding and edge with an angle of 90 degrees.
	View Visualization	Visualize all four business process modeling views in separate modeling windows (i.e., for each view one window), but enable the simultaneous presentation and editing of all views.
	Decomposition Visualization	The tool should comprise notification mechanisms, indicating to the modeler that he or she is currently not working on the highest decomposition level. Hence, if the modeler decomposes a business object or business transaction, the corresponding elements in the Interaction Schema and Task-Event Schema should indicate this information at the decomposed element.
	Font size, window-zooming, TES layout	The modeler should be able to customize the visualization of the SOM models by means of changing e.g., the background color of the elements or the fonts of the labels. Moreover, it should be customizable, whether the tool automatically adjusts the windows width and height after each modeling operation execution. For the TES window, it should be customizable which business object tasks are assigned to which vertical position, hereby increasing readability.

Validation	Validate Model	Provide validation functionality by means of checking the consistency between the multiple SOM modeling views.

An interesting non-functional requirement that is typical for multi-view modeling of SOM business process models is the differentiation between zooming and the definition of new business process model levels. The SOM tool should enable the definition of different levels of detail of the business process. Zooming between already defined levels is targeting at providing navigation functionality and obtaining overview of large business process models. By contrast, with the *increase* and *decrease* operators, the modeler should be enabled to define higher or revoke the current level of the business process model, respectively.

7.2.6 Step VI: Evaluation

The MUVIEMOT lifecycle is concluded with the *Evaluation* step. The method does not specify the evaluation approach generally, the goal is to evaluate the conceptual design according to completeness, utility and usability. In the following section, the conceptual design is evaluated by a technical realization. Therefore, the requirements guided the development of a SOM multi-view modeling tool based on the ADOxx meta modeling platform.

7.3 Development of a SOM Multi-View Modeling Tool on ADOxx

In this section, the conceptual design of a multi-view modeling tool for the SOM method, as defined in section 7.2.5, is used to guide the development of a corresponding modeling tool. The SOM tool has been realized with the ADOxx meta modeling platform. It is available for download on the SOM project page within the Open Models Initiative (OMI) (KARAGIANNIS ET AL., 2008). The following sections describe the application of the conceptual design in a concrete realization, utilizing the functionality of the ADOxx meta modeling platform.

7.3.1 The ADOxx Meta Modeling Platform

Today, there are several frameworks (e.g., *Eclipse Graphical Modeling Project* [32] that uses *Eclipse Modeling Framework* [33] and *Eclipse Graphical Editing Framework (GEF)* [34] to build

[32] `http://www.eclipse.org/modeling/gmp/`, last checked: 2013-07-26
[33] `http://www.eclipse.org/modeling/emf/`, last checked: 2013-07-26
[34] `http://www.eclipse.org/gef/`, last checked: 2013-07-26

generative components and runtime infrastructure for graphical editors) and meta modeling platforms (e.g., *ADOxx*[35], *MetaEdit+* (KELLY ET AL., 1996; TOLVANEN AND ROSSI, 2003)) that can be used to develop graphical modeling editors. Because of the broad and successful utilization of ADOxx in the development of manifold modeling tools originating from a scientific background (e.g., (FILL, 2005, 2012; FILL ET AL., 2013; KARAGIANNIS AND TELESKO, 2000; LICHKA ET AL., 2002; SCHWAB ET AL., 2010)), and the combination with the Open Models Intiative (OMI)[36], the decision was made to use ADOxx as a development platform. The Open Models Initiative is dedicated to establish a community of modelers with both, research and practical background. All tools developed within the project can be used free of charge, hereby fostering the diversity and the diffusion rate of modeling methods.

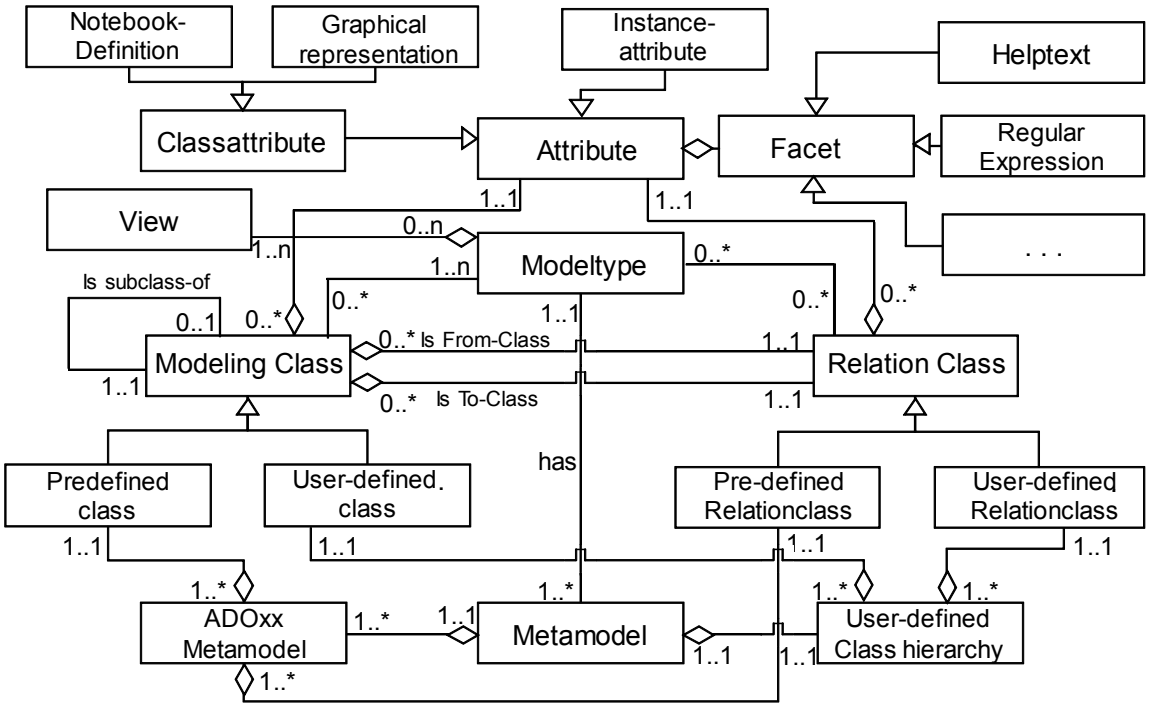


Figure 55: Excerpt of the ADOxx meta meta model (cf. (FILL AND KARAGIANNIS, 2013; KÜHN ET AL., 1999))

ADOxx is a meta modeling platform developed by the BOC AG[37], a spin-off of the University of Vienna. The BOC was founded 1995 by a team led by Prof. Dr. Dimitris Karagiannis. The first product of the BOC was a business process modeling tool called *ADONIS* (JUNGINGER ET AL., 2000). Over the years, several additional products, i.e., modeling tools have been developed. In order to improve efficiency, a meta modeling approach has emerged for modeling tool development. Hence, the focus was on capturing the generic constituents of several created

[35] http://www.adoxx.org/live/, last checked: 2013-07-26
[36] http://www.openmodels.at (KARAGIANNIS ET AL., 2008), last checked: 2015-02-09
[37] http://www.boc-group.com/, last checked: 2015-02-09

modeling tools into one meta modeling approach. ADOxx was born. ADOxx provides a generic meta meta model that can be instantiated, i.e., customized, to the specific needs of the modeling method at hand. Moreover, based on this meta meta model, a generic platform-specific meta model is available. This meta model is used to realize platform-specific functionality like analysis, pre-defined model queries, and model simulation. Moreover, the platform comes with pre-configured functionality for storage and manipulation of graphical models in relational databases, multi-user access and model validation.

The foundation of the ADOxx platform is the meta meta model illustrated in Figure 55. The platform is built on a layered architecture, using the established four meta layers: *original*, *model*, *meta model*, and *meta meta model* that are used in other popular approaches like EMF or the MOF. Figure 56 illustrates the modeling hierarchy and assigns the corresponding roles, thus, describing the realization of user-specific meta models with ADOxx (as an instantiation of the generic meta modeling framework illustrated in Figure 9). Using this ADOxx meta model, a *Meta Modeler* instantiates his or her *User specific Meta Model*. This meta model is stored in a platform-specific language called *ALL*. A *Modeler* then creates valid *Models* as instances of the user specific meta model. ADOxx stores these models in *ADL/XML*, a platform-specific but XML-based notation. This modeling hierarchy and the generic implementation of the platform functionality on the ADOxx meta meta model and ADOxx meta model enable automatic generation of domain-specific modeling tools with built-in functionality.

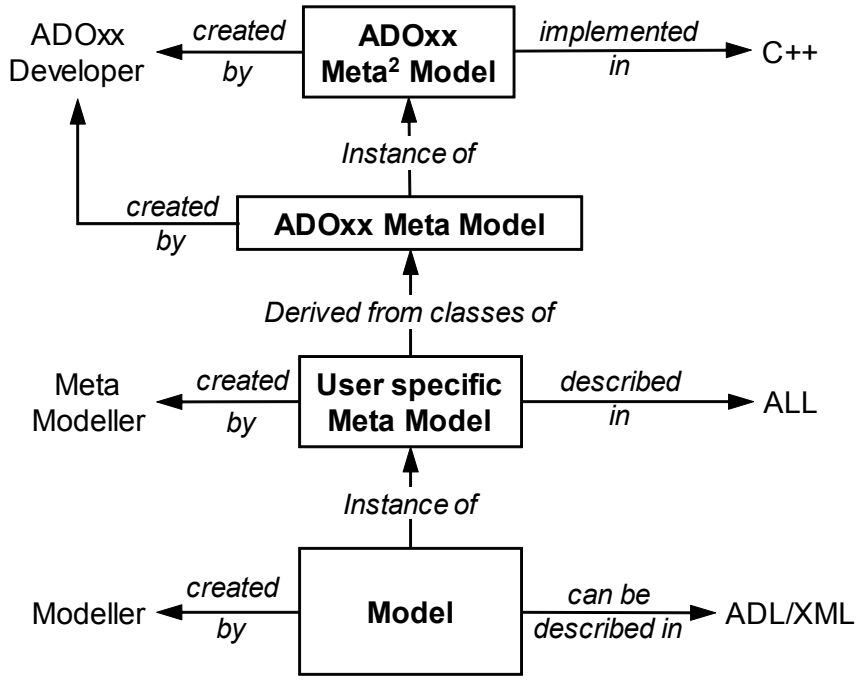


Figure 56: Roles and languages in the modeling hierarchy of ADOxx (FILL AND KARAGIANNIS, 2013)

7.3.2 The SOM Multi-View Modeling Tool Development

The first step in the development of modeling tools based on ADOxx is the specification of the user specific meta models. These meta models need to be defined by relating the user specific meta model concepts to the concepts of the platforms meta model by means of inheritance relationships. The choice of the super class determines platform functionality that is also inherited by the sub class, i.e., the ADOxx class *__D_Event__* and its sub classes come with pre-defined simulation attributes that can be customized to any modeling method with process-oriented execution semantics. An example for the visualization is the ADOxx class *__D_Container__*. All sub classes of this class inherit the operation *is_inside*. This functionality provides the meta modeler with all object ids that are inside an instance of the __D_Container__ class.

7.3.2.1 Realization of the SOM Viewpoints

As the conceptual design defines, the SOM business process modeling tool comprises four viewpoints. Moreover, on the resource layer, two more SOM viewpoints are defined. The BPMN viewpoint adds an alternative viewpoint to the resource layer. The ADOxx platform provides the concepts of *modeltypes* (cf. Figure 55). A modeltype, most importantly, comprises a set of *Modeling Classes* and *Relation Classes*, hereby realizing the modeling language provided within an ADOxx model. Modeling classes can be used for any node of graphical modeling languages, e.g., activities in BPMN. Relation classes realize edges between the nodes, e.g., sequence flows in BPMN. Hence, the first step in developing a SOM multi-view modeling tool was to map the concepts of the SOM meta models to the concepts of the ADOxx meta model. Thereafter, the multiple ADOxx modeltypes can be specified by explicitly including the concept that should be considered in a modeltype (i.e., a viewpoint).

Table 24: Mapping of the SOM meta model concepts to the ADOxx meta model concepts

ADOxx meta model	SOM meta model
Modeling Class	Environmental Object, Object of Discourse, Business Transaction (in TDS), Task, Object Specific Objecttype, Service Specific Objecttype, Transaction Specific Objecttype, Task Specific Objecttype
Relation Class	Initiating Transaction, Contracting Transaction, Enforcing Transaction, Control Transaction, Report Transaction, External Event, Internal Event, Object Decomposition, Transaction Decomposition, is_a, interacts_with, is_part_of

Table 24 shows the mapping between the SOM meta model concepts and the ADOxx mod-

eling classes and relation classes. Notably, business transactions are realized by a modeling class and a relation class, depending on the viewpoint considered: First, in the Transaction Decomposition Schema, business transactions are realized as modeling classes, i.e., nodes, in the hierarchical decomposition tree, related to each other by decomposition relationships. The specific types of business transactions, i.e., initiating, contracting etc.. is specified by an enumeration attribute type. Second, in the Interaction Schema and Task-Event Schema, business transactions are realized as relation classes, i.e., as edges, connecting two tasks. In the latter case, only the specific business transactions are visualized. They can be distinguished by their color, e.g., enforcing transactions are colored green.

After the meta model mapping has been performed, modeltypes have been specified to group modeling classes and relation classes, hereby realizing the multiple viewpoints of SOM in ADOxx. Table 25 illustrates the modeltypes and the SOM meta model concepts considered by them. The modeltypes are later used by the modeler to actually create new models according to a SOM viewpoint.

Table 25: Mapping of the SOM meta model concepts to the ADOxx modeltypes

ADOxx modeltype	SOM Modeling Classes and Relation Classes
Transaction Decomposition Schema	Business Transaction
	Transaction Decomposition
Object Decomposition Schema	Environmental Object, Object of Discourse, Business Transaction
	Object Decomposition, Transaction Decomposition
Interaction Schema	Environmental Object, Object of Discourse
	Initiating Transaction, Contracting Transaction, Enforcing Transaction, Control Transaction, Report Transaction
Task-Event Schema	Task
	Initiating Transaction, Contracting Transaction, Enforcing Transaction, Control Transaction, Report Transaction, Internal Event, External Event

Schema of Task Classes	Task Specific Objecttype
	interacts_with
Schema of Conceptual Classes	Object Specific Objecttype, Service Specific Objecttype, Transaction Specific Objecttype
	is_a, interacts_with, is_part_of

The realization of the viewpoints as ADOxx modeltypes is exemplified for the Task-Event Schema in Figure 57[38]. The modeltype specification must follow a specific syntax. First, the keyword *MODELTYPE* indicates the start of a new modeltype. For each modeltype certain parameters can be specified, e.g., whether it enhances an already defined modeltype (parameter *from*), defining the icon of the modeltype visualized in the model explorer (keyword *bitmap*), or defining the attributes of the modeltype (keyword *attrrep*). Subsequently, all modeling and relation classes are added to the modeltype using the keyword *INCL* followed by the name of the class.

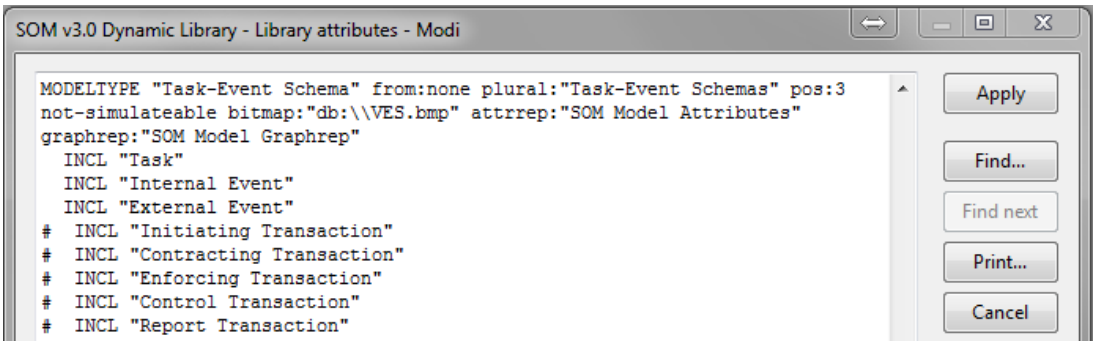


Figure 57: Task-Event Schema modeltype specification in ADOxx

Comparing Figure 57 with the corresponding row in Table 25, one may wonder, why all business transactions, e.g., initiating transaction and contracting transaction, are not specifically included in the modeltype - the # is used to comment out the code of the whole line. The reason is, that the SOM method does not support drag & drop modeling. Instead, model editing is performed by recursively applying the decomposition rules and increasing/decreasing the business process level. Hence, all tasks and business transactions of the TES can be automatically derived from the decomposition trees and the Interaction Schema. Not including the

[38] For readability and consistency reasons, the names of the concepts have been translated into English although originally, they have been implemented in German on the platform.

SOM Model Attributes - Edit facets - Standard value

```
NOTEBOOK

CHAPTER "Visualisation" [1]
GROUP "Model Visualisation" [2]
ATTR "Visualised Views" [3]
ATTR "Define Window Arrangement" no-param push-button [4]
ENDGROUP [2]

GROUP "Context Visualisation"
ATTR "Models visualising Context"
ENDGROUP

GROUP "Apply Auto-Layouting for the following Views "
ATTR "Auto-layouted Views"
ENDGROUP

ATTR "Task-Event Schema Visualisation" no-param push-button

ATTR "Theme"

ATTR "font size"

ATTR "Window Zooming"
```

Figure 58: AttrRep definition

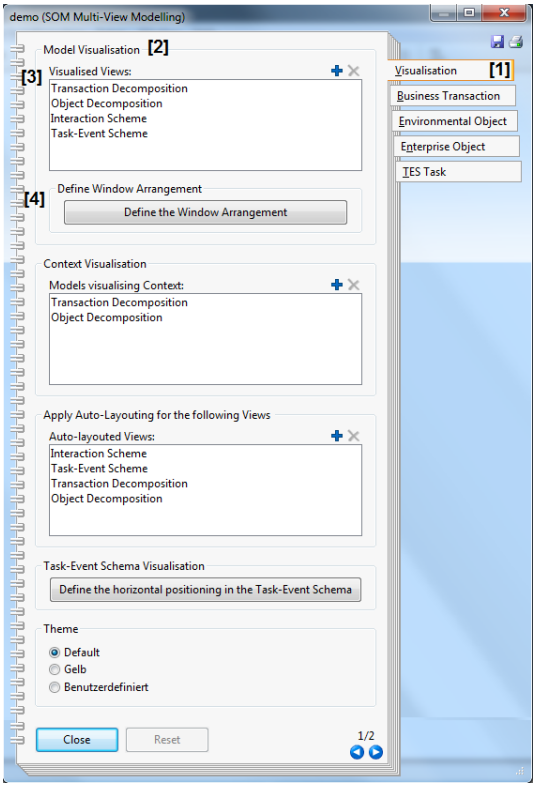


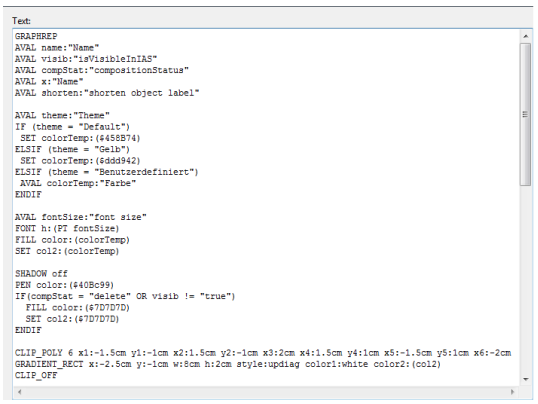
Figure 59: AttrRep visualization

business transactions in the modeltype enables the creation of instances via script, but not by the modeler using drag & drop functionality. The palette of modeling concepts for TES models will not provide these concepts to the modeler.

As mentioned already, the ADOxx meta modeling platform comes with built-in functionality that is realized for the new created modeling methods by inheritance of the ADOxx meta model concepts. With these concepts, platform-specific attributes are also inherited and need to be specified for the new user-specific meta model concepts. The most important attributes are now described briefly:

AttrRep Attributes of modeling classes, relation classes and modeltypes are presented to the modeler for editing purposes by means of an ADOxx *Notebook*. This Notebook can be specified in the class attribute value of the attribute *AttrRep* (an abbreviation for attribute representation). The ADOxx platform provides a specific syntax for describing Notebooks. The meta modeler can decide which attributes should be visible to the modeler and which attributes are editable by the modeler. The platform provides different, predefined attribute representation types, e.g., check boxes, radio buttons, and lists. Figure 58 illustrates the implementation of the ADOxx Notebook for the modeling area used in any SOM business process model modeltype.

After the keyword *NOTEBOOK* follows the actual code for the attribute visualization.



```
Text:
GRAPHREP
AVAL name:"Name"
AVAL visib:"isVisibleInIAS"
AVAL compStat:"compositionStatus"
AVAL x:"Name"
AVAL shorten:"shorten object label"

AVAL theme:"Theme"
IF (theme = "Default")
 SET colorTemp:($458B74)
ELSIF (theme = "Gelb")
 SET colorTemp:($ddd942)
ELSIF (theme = "Benutzerdefiniert")
 AVAL colorTemp:"Farbe"
ENDIF

AVAL fontSize:"font size"
FONT h:(PT fontSize)
FILL color:(colorTemp)
SET col2:(colorTemp)

SHADOW off
PEN color:($40Bc99)
IF(compStat = "delete" OR visib != "true")
  FILL color:($7D7D7D)
  SET col2:($7D7D7D)
ENDIF

CLIP_POLY 6 x1:-1.5cm y1:-1cm x2:1.5cm y2:-1cm x3:2cm x4:1.5cm y4:1cm x5:-1.5cm y5:1cm x6:-2cm
GRADIENT_RECT x:-2.5cm y:-1cm w:8cm h:2cm style:updiag color1:white color2:(col2)
CLIP_OFF
```

Figure 60: Excerpt of the business transaction GraphRep code

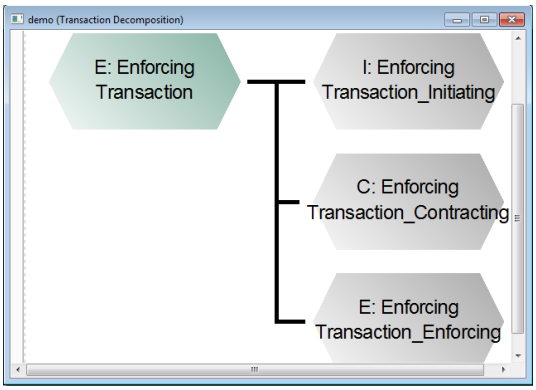


Figure 61: Business transaction visualization

The keyword *CHAPTER* is used to define a top-level structuring of the notebook (see [1] in Figure 58). Within a chapter, the next structuring element is realized by the keywords *GROUP* and *ENDGROUP*, resulting in a surrounding box with all attributes defined within the two keywords (see [2] in Figure 58). All attributes that should be included in a Notebook must be specifically included with the keyword *ATTR* followed by the name of the attribute in parenthesis. Optionally, permitting the change of the attribute values by the modeler or defining the representation type, e.g., *no-param push-button* for the attribute "*Define Window Arrangement*" (see [3] and [4] in Figure 58) can be implemented. Figure 59 shows the resulting visualization of the Notebook in the SOM business process modeling tool.

GraphRep Each Modeling Class and Relation Class needs to have a notation in order to be visualized in the modeling area and to be interpretable by human beings. This notation plays a mature role for efficiency and usability of the modeling tool. Using the class attribute *GraphRep*, the meta modeler is able to define the graphical representation for each meta model concept using a platform-specific language. Figure 60 shows a snippet of the platform-specific code used for the definition of the graphical representation of *Business Transactions* in the Transaction Decomposition Schema.

After the keyword *GRAPHREP* follows the actual code for the graphical visualization. The platform provides the meta modeler with the possibility to adopt the notation of a modeled instance immediately depending on changes of attribute values. Using the keyword *AVAL* followed by the name of an attribute in parenthesis enables the tool to recognize attribute value changes and trigger corresponding changes of the visualization (see [1] in Figure 60). The example in Figure 60 shows the attribute-dependent fill color of the oval, i.e., grey, if the *compositionStatus* attribute value is *delete* or *visible* = false; or green otherwise.

```
Text:
GRAPHREP
SHADOW off
AVAL x:"Name"
AVAL shorten:"shorten transaction label"
AVAL linebreak:"Linebreak"
AVAL fontSize:"font size"
AVAL conApp:"Context Appearance"
SET fontSize:(PT fontSize)
SET penCol:"dodgerblue"

IF (shorten = "Yes" AND (LEN x > 17)) {
  SET x:(copy(x, 0, 17) + "...")
}

PEN color:dodgerblue w:.08cm
AVAL visib:"isVisibleInIAS"
IF(visib = "delete")
  PEN color:gray w:.08cm
ENDIF

EDGE

AVAL o:"Text Orientation"
FONT "Arial" h:(fontSize) color:dodgerblue line-orientation:(VAL o)
AVAL triggerDec:"triggerDecomp"
IF((VAL triggerDec) = 1) {
FONT "Arial" h:(fontSize) color:dodgerblue line-orientation:(VAL o)
underline italic bold
}
```

[1] [2]

Figure 62: Excerpt of the contracting transaction GraphRep code

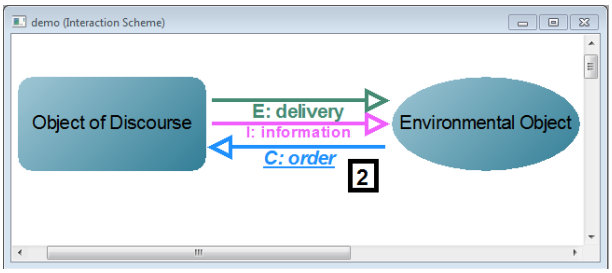


Figure 63: Contracting transaction visualization

Figure 61 visualizes the attribute-dependent visualization using a sample Transaction Decomposition Schema. The first enforcing transaction has a composition status of *visible*, i.e., it is part of the currently visualized business process level in the Interaction Schema and Task-Event Schema. Consequently, the other three business transactions are currently not visible in IAS and TES, therefore they are filled with grey color.

Figure 62 and Figure 63 illustrate the implementation of the notation of the business transaction relation and the corresponding visualization, respectively. First, several attribute values are stored in local variables using the *AVAL* operator. The if-statement ([1] in Figure 62) checks, whether the variable *shorten transaction label* is set to *"Yes"*. If that is the case, then the name of the transaction will be shortened to 17 characters, followed by three dots. The third if-statement ([2] in Figure 62) is responsible for visualizing the name of the business transaction in italics, underlined and bold if the attribute value of *triggerDecomp* is equal to *1*. This internal attribute is modified only using AdoScript. Its default value is 0. The value is only set to 1, if the modeler has decomposed the business transaction. Figure 63 ([2]) illustrates this case for the contracting transaction *C: order*.

Class Cardinality This attribute is part of the ADOxx root classes *__D|S_Construct__*. Therefore, this attribute is inherited by any relation class and modeling class introduced to the platform. As indicated by the name, this attribute can be used to constrain the ingoing and outgoing relationships of a class, specified for any specific relationship type (i.e., ADOxx relation class). Moreover, minimal and maximal number of instances of a class in any model can be specified. The platform can be customized to check this constraints any time the modeler tries to save a model or only when the functionality is manually trig-

gered in the menu. ADOxx provides a specialized syntax for defining such constraints in a straight forward manner.

In case of the SOM modeling tool, class cardinalities have been used, e.g., to restrict the number of incoming interacts_with relationships to the instances of the modeling class object specific Objecttype to 0. Hence, object specific Objecttypes are not allowed to have a incoming relationship of this type.

Model pointer The last platform-specific attribute that is important for the realization of the SOM modeling tool are the model pointers. With this special attribute type, the meta modeler is enabled to specify an attribute for a class, whose attribute value is a reference to either another model as a whole, or to a certain modeling class or relation class in a certain model. This reference allows the modeler to jump between the two referenced elements and switch to a different model via one mouse click.

In case of the SOM method, these model pointers have been used, e.g., to realize the relationships between business transactions and business objects in the decomposition schemata to the corresponding tasks in the Task-Event Schema. Moreover, references have been established to identify the derivation of task specific objecttypes in the Schema of Task Classes from there corresponding tasks in the Task-Event Schema.

7.3.2.2 Realization of the SOM Business Process Modeling Procedure

The SOM method utilizes multi-view modeling by design. Therefore, it was required to visualize all four SOM viewpoints to the modeler simultaneously. Figure 64 shows the visualization of an initial SOM business process model in the ADOxx modeling tool. On the most left side is the *Model Groups Explorer* that visualizes all models within the model groups as well as the *Navigator* (on the bottom), that can be used to adjust the visualized area of the model. In the center of the tool is the modeling area. The tool is implemented to divide the modeling area into four distinct model types, each representing one viewpoint of the SOM business process model (BORK AND SINZ, 2010, 2011a): the Transaction Decomposition Schema on the upper left side, the Object Decomposition Schema on the upper right side, the Interaction Schema on the lower left side, and the Task-Event Schema on the lower right side of the modeling area.

In order to refine this initial SOM business process model accordingly to the system under study, the modeler has to apply the SOM decomposition rules (see Table 17). These rules have been implemented consistently by integrating them into the context menu of the modeling classes and relation classes they can be applied on. Consequently, after performing a right-click on a business object, the modeler can select the operator *Decompose Object* followed by the selection of one of the applicable decomposition rules for object decomposition. The

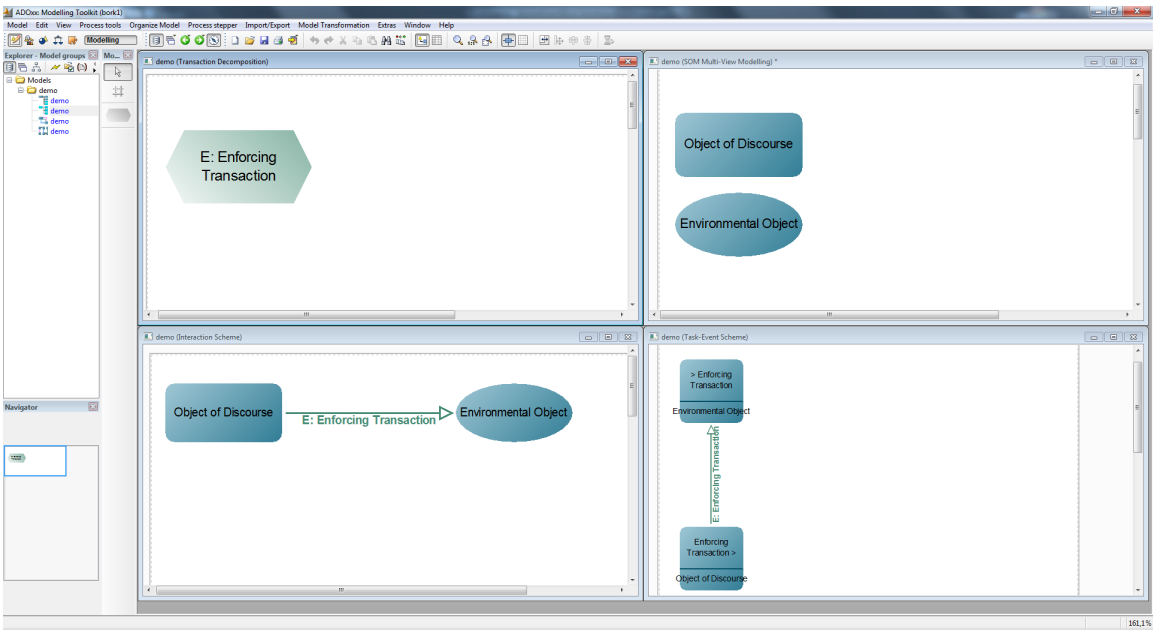


Figure 64: Initial screen of the SOM business process modeling tool

same procedure is implemented for the decomposition of business transactions and the zooming operator.

Adding and removing of additional enforcing transactions or environmental objects to the initial SOM business process model is utilized by corresponding operators integrated into the menu bar of the SOM modeling tool. As they affect the whole business process, it was considered not appropriate to attach these operators only to certain modeling concepts or viewpoints.

The only scenario, the modeler utilizes drag & drop modeling is when he or she defines the behavior of the business process. Following the SOM method, the behavior is specified by relating two tasks, belonging to the same business object, with an internal event relationship; or by connecting an external event with a task. Consequently, both modeling actions can only be performed in the Task-Event Schema. The modeler selects the relationship type in the list of provided relationship types and consecutively clicks on the tasks, the relationship is outgoing and incoming. A comprehensive documentation can be found on the SOM tool's homepage[39], supplemented with tutorial videos showcasing the usage of the tool.

The model-driven derivation of the resource layer models based on SOM business process models is triggered using the menu bar of the SOM tool. In the menu *Model Transformation*, the modeler can select between *Generate Schema of Task Classes*, *Generate Schema of Conceptual Classes*, and *Generate BPMN*. Prior to each transformation, mechanisms check, whether a valid SOM business process model is given, i.e., checking whether the whole behavior of the business process is defined in the TES or not. If the check is passed, the AdoScript algorithms perform

[39] The SOM project page: `http://www.omilab.org/web/som`, last checked: 2015-02-13

the meta model based transformation of the currently visualized SOM business process model into the specified target modeltype. The new created model is automatically integrated into the model group of the business process model it is derived from.

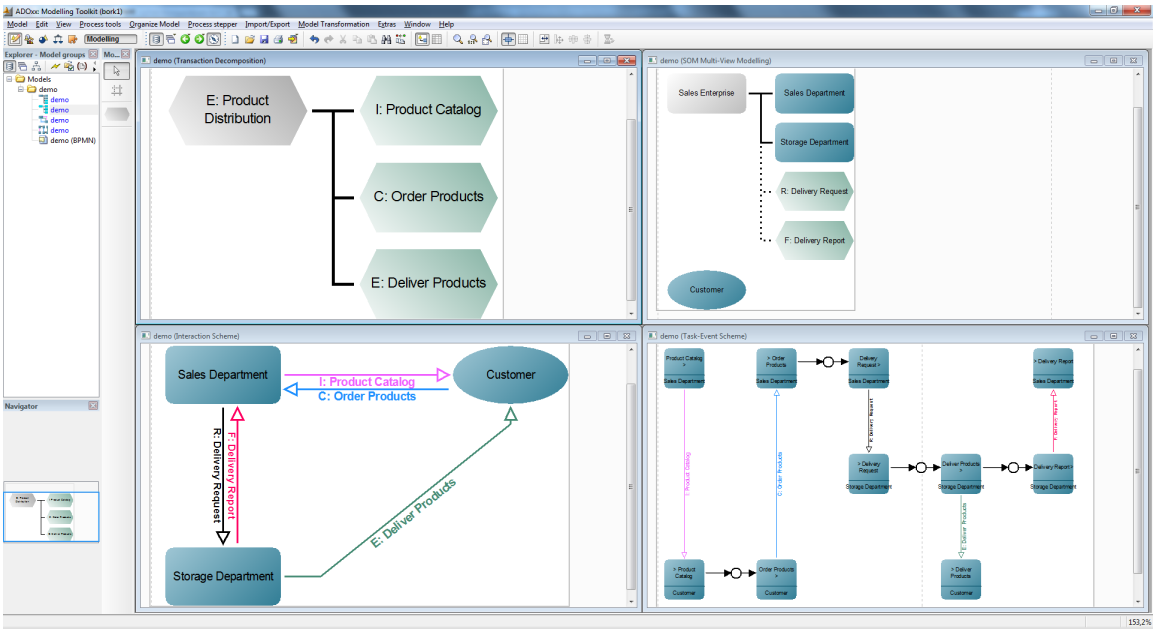


Figure 65: SOM business process modeling with ADOxx

Figure 65 shows a SOM business process model for a common product selling enterprise. The initial business process model has been decomposed two times: First, the initial product distribution transaction has been decomposed according to the negotiation principle into a sequence of three business transactions (*Product Catalog*, *Order Products*, and *Deliver Products*). Second, the initial sales enterprise business object has been decomposed according to the *feedback and control principle* into two business objects, *Sales Department* and *Storage Department*, with a hierarchical coordination by control and report transactions *Delivery Request* and *Delivery Report*, respectively.

In a subsequently performed modeling step, the behavior has been defined within the Task-Event Schema. The process is initiated with a product catalog that is sent to the customer. The customer then creates an order. The receipt of this order from the sales department causes a control transaction, transferring the order and the letter information to the storage department. The storage department then sends the products to the customer followed by a report transaction to the sales department in order to report on the carried out task.

Figure 65 also visualizes the result of the attempt to foster usability of the tool in multiple ways: First, algorithms have been implemented that are responsible for the automatic layout of the SOM models every time the user has performed a modeling operation. Hence, the position of the business objects and the routing of the business transactions visualized in all viewpoints

of the SOM example have not been modified before the screen shot, visualized in Figure 65, has been taken. This significantly improves efficiency and usability of the tool. In this regard, also auto-layout algorithms for the Schema of Task Classes, the Schema of Conceptual Classes, and the Business Process Modeling and Notation models have been implemented.

Second, looking at the decomposition schemata, it can be seen, that the initial business transaction *product distribution* as well as the initial object of discourse *sales enterprise* are visualized with grey color, whereas their respective decomposition products are colored green and blue. The color-coding is utilized to enable the modeler to immediately identify the business objects and business transactions that are part of the currently visualized business process level in Interaction Schema and Task-Event Schema.

Third, the initial positioning of the four viewpoints is automatically generated during the initialization of a new SOM business process model. The tool checks the resolution of the currently used computer screen and divides the available space for the modeling area into the four viewpoints to be visualized simultaneously. Notably, this list does not claim to be complete, plenty of functionality targeting at usability of the tool, e.g., customization of the color, the fonts, the positioning and the visualization of viewpoints, are not described for brevity reasons.

7.3.2.3 Realization of the SOM Viewpoint Dependencies

The Viewpoint Dependencies specified in the conceptual design have been realized in manifold AdoScript lines of code and attributes that are not visible to the modeler, but needed by the tool in order to keep the multiple views consistent. For example, the relationship between a task and a business object is realized by an ADOxx model pointer. Two examples of hidden attributes might help to gain some insights on the benefits of the conceptual design during the implementation step. As business transactions are visualized by modeling classes and relation classes, depending on the SOM viewpoint, an attribute is attached to any created instance that has a list of all object ids of this instance, i.e., ids of other modeling elements in the different SOM views that visualize the very same instance. After attribute changes have been performed by the modeler, AdoScript code is executed that iterates over this id list and changes the attribute values of these elements to the current attribute value.

Second, the behavior of the business process model defined by the modeler is stored in the decomposition schemata. The decomposition schemata serves as an integrated model of the whole SOM business process model. Any time the modeler applies the zooming operator, the Interaction Schema and Task-Event Schema might change completely. However, before changing the visualized business process level, the tool stores the behavior information in the corresponding business objects and business transactions of the decomposition schemata. Hence, if the modeler applies the zooming operator again, the tool can retrieve the behavior information and automatically create the internal events and external events in the Task-Event Schema.

7.3.2.4 Realization of the SOM Resource Layer Viewpoints

The SOM resource layer is comprised of the viewpoints: Schema of Task Classes and Schema of Conceptual Classes. Moreover, the model-driven transformation of SOM business process models into Business Process Modeling and Notation (BPMN) workflow schemata, defined by PÜTZ AND SINZ (2010a,b), can be considered a third viewpoint on this layer. The SOM business process model of Figure 65 is used in the following to briefly describe the different model transformation approaches.

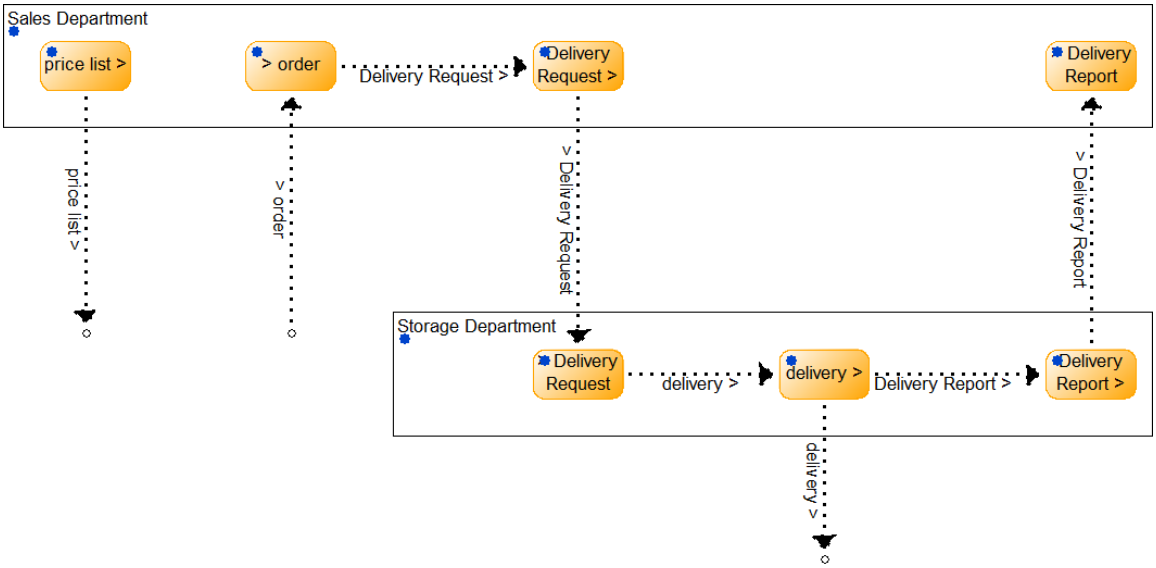


Figure 66: Derived Schema of Task Classes in the SOM tool

Figure 66 shows an excerpt of the generated Schema of Task Classes (TAS). The figure shows the task specific objecttypes for the two business objects *Sales Department* and *Storage Department*. The modeler can select which business objects shall be considered during the transformation. As specified by the meta model mapping in section 7.1.3.2, each task of the Task-Event Schema has been transformed into a task specific objecttype in the TAS. The objecttypes are connected by *interacts_with* relationships, derived from the internal event relationships of the corresponding tasks of the TES. Incoming and outgoing connections from and to business objects are connected with a dummy endpoint as the platform needs two classes, incoming and outgoing, for each relation class. This is only necessary, if the related task specific objecttype is connected with a business object not considered during the transformation. Hence, if all business objects are selected by the modeler, no dummy endpoints are required in the TAS.

Similarly to the business process modeling viewpoints, also the resource layer viewpoints provide auto-layout algorithms. In case of the Schema of Task Classes, all task specific objecttypes are aggregated in the corresponding bounding box of the business object they are derived

from. Ferstl and Sinz refer to this bounding box and the included objecttypes as the Schema of Task Classes. Consequently, the model visualized in Figure 66 shows two TAS - not one. Positioning of all objecttypes and relationships are automatically computed with the goal of prevent intersections and drawing all relations directly.

As discussed in section 7.1.3.2, SOM defines several modeling operations that can be applied to further refine the initially derived Schema of Task Classes. In the example visualized in Figure 66, the modeler is enabled to delete task specific objecttypes if the execution of the corresponding tasks are not automatable (e.g., sending *Product Catalog* >). Also, the modeler can merge task specific objecttypes whose corresponding tasks have to be performed corporately every time (e.g., merging of the objecttypes > *Order Products* and *Delivery Request* > into a compound objecttype *Order Processing*). Finally, the modeler can define the message formats and operations for each task specific objecttype. A complete list of further modeling operations can be found in (FERSTL AND SINZ, 2013, p. 233) and section 7.1.3.2. The tool supports all proposed refinement operations.

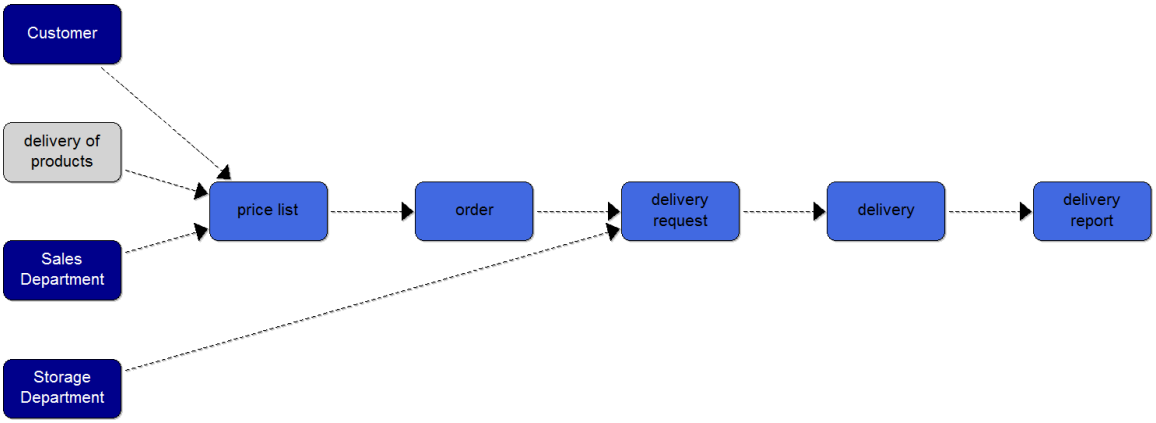


Figure 67: Derived Schema of Conceptual Classes in the SOM tool

Figure 67 shows the initially derived Schema of Conceptual Classes (COS). The COS is derived from the same business process model as the TAS (see Figure 65). The transformation is based on the specified meta model mapping (see section 7.1.3.1). Each business object, i.e., Customer, Sales Department, and Storage Department, has been transformed into an object specific objecttype, positioned on the left border of the model. The initial enforcing transaction *distribution of products* is transformed into a service specific objecttype, also positioned on the left border of the model. Each business transaction connected by two business objects in the business process model is transformed into a transaction specific objecttype, connected with the respective object specific objecttypes in the COS. The sequence of transaction specific objecttypes is determined by the behavior of the business process model specified in the Task-Event Schema. Therefore, a valid and comprehensive transformation of a Schema of Task Classes considerably depends on the complete specification of the business process behavior.

The positioning of the objecttypes is automatically performed by an auto-layout algorithm implemented for COS models. Independent objecttypes, i.e., object specific objecttypes and service specific objecttypes are positioned on the left border, whereas task specific objecttypes are drawn from left to right, depending on the behavior specification of the business process model. The modeler can further change the initial layout by iterating the different possibilities of positioning the independent objecttypes on the left border and by aligning the task specific objecttypes either on the top of the modeling area or in the center (as shown in Figure 67).

As has been mentioned in section 7.1.3.1, several refinement operations can be applied to the initially derived COS. In the example illustrated in Figure 67 the modeler could in a first step e.g., delete the *Product Catalog* objecttype because the execution of the corresponding tasks is not automatable; or the modeler could introduce a normalization for the *Order Products* objecttype, hereby creating an *Order* objecttype and an *Order Item* objecttype related to each other by a normalization relationship. Moreover, the modeler can define the message formats and operators for each objecttype. A complete list of further modeling steps can be found in (FERSTL AND SINZ, 2013, p. 229f.) and section 7.1.3.1.

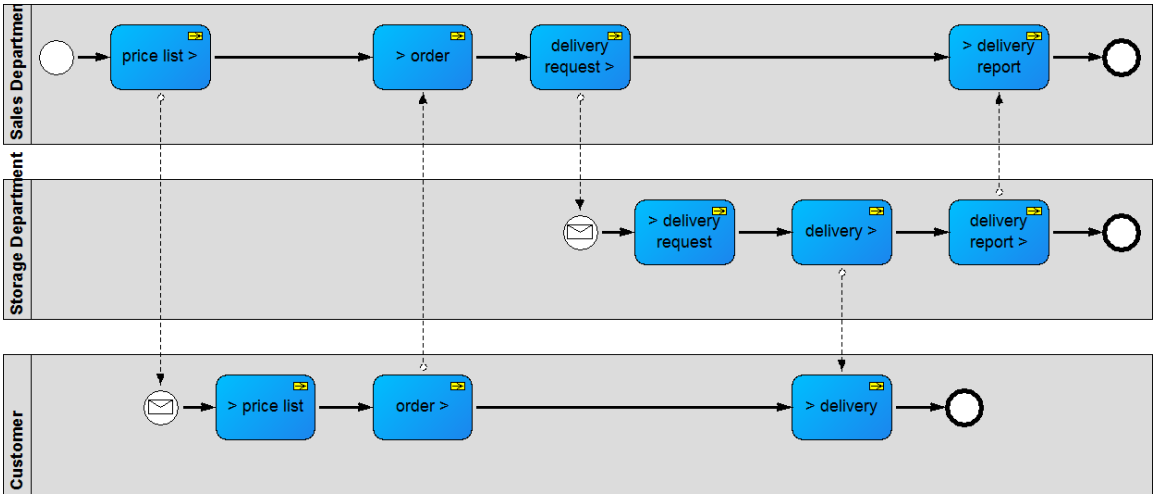


Figure 68: Derived BPMN workflow schemata in the SOM tool

Figure 68 shows the initially derived workflow specification by means of a Business Process Model and Notation (BPMN) model. The generation of the BPMN model is based on the meta model mapping defined by PÜTZ AND SINZ (2010a) (see section 7.1.3.3 for a complete description of the mapping). Each business object involved in the business process is transformed into a *pool*, i.e., Sales Department, Storage Department, and Customer. Each task in the Task-Event Schema is transformed into an *activity* in the BPMN model, e.g., Order Products >, >Delivery Request, or >Deliver Products. If two consecutive tasks belong to the same business object, i.e., there exists and internal event between these tasks in the TES, a *sequence flow* object is created and connected related between the two activities, e.g., between >*Order Products* and

Delivery Request >. If two consecutive tasks belong to different business objects, i.e., they are connected by a business transaction in the TES, a *Message Flow* object is created and related between these two activities, e.g., between *Order Products* > and >*Order Products*. One *start event* is created and connected with the activity corresponding to the first task in the TES, i.e., *Product Catalog* >. *End Events* are also created and added to the end of each pool, e.g., after >*Deliver Products*.

The model visualized in Figure 68 is corresponding to the initially transformed and layouted BPMN model. The SOM tool provides a complex auto-layout algorithm that ensures a proper visualization of the BPMN models. The algorithm pursues the goal of preventing intersections. Activities are horizontally aligned and vertically centered within their corresponding pool. Their horizontal position is determined by the position of the activity they are connected with. If the other activity is part of the same pool, a horizontal default empty space is added, otherwise, if the connected activity is assigned a different pool, the current activity is positioned directly above or beyond that activity in order to allow a direct relation.

The initially derived BPMN model can then be further processed with the tool. One refinement is the creation of *subprocesses* by selecting activities within a pool, executing the *Merge to Subprocess* operator in the context menu, and assigning a name for the new subprocess. The tool then visualizes a surrounding box around the selected activities in the pool together with the name of the subprocess centered on top of the surrounding box. Another refinement is the extraction of one or more activities into a new pool by selecting the activities and executing the *Move to new Pool* operator in the context menu of the activities. Activities that correspond to tasks that are not automatable can be removed from the BPMN schemata, and activities not yet considered in the transformed TES can be added. While removing an activity, the tool checks, with which other activities the selected activity has been connected by a sequence flow relationship. After the deletion is performed, an algorithm automatically routes the incoming relationships of the deleted activity to the endpoint activity of the outgoing relationships, i.e., if activities A, B, and C are connected by a sequence flow relationship (r), A r B r C, and the modeler deletes activity B, the BPMN model will show activity A connected with activity C by a sequence flow relationship, A r C.

The modeler has two possibilities for adding new activities: (1) dropping an activity into a pool, or (2) dropping an activity directly on a sequence flow relationship. Selecting option (1), the modeler has to manually add a sequence flow relationship, whereas option (2) automatically integrates the new activity into the selected sequence flow. Figures 69 and 70 illustrate an excerpt of an BPMN model prior (Figure 69) and after (Figure 70) using the option (2) of adding a new activity.

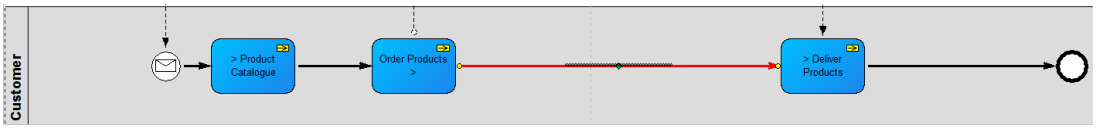


Figure 69: Adding a new activity element in BPMN -1-

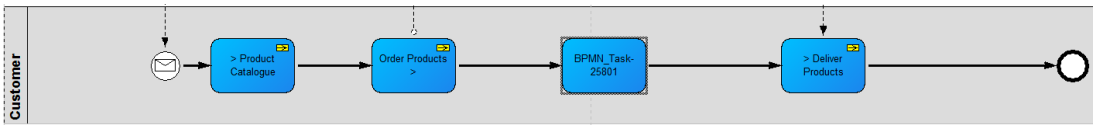


Figure 70: Adding a new activity element in BPMN -2-

7.3.2.5 Realization of the SOM Mechanisms & Algorithms

In the following, a brief description of the most important mechanisms and algorithms of the SOM modeling tool is given. The requirement for these additional functionality comes from usability considerations, in case of the layout algorithms; and from the Viewpoint Dependencies, otherwise.

Business Process Viewpoints Synchronization Mechanisms The four viewpoints of the SOM business process layer need to be kept consistent at all times. As the Viewpoint Dependencies specify, there are manifold relationships between modeling concepts of different viewpoints. On the one hand, several SOM modeling concepts like business transactions and business objects are used in multiple viewpoints. On the other hand, attribute values of some modeling concepts depend on attribute values of other modeling concepts, e.g., the corresponding object and of a task or the connected transaction of a task.

It was therefore vital for the SOM modeling tool to capture these dependencies in an algorithmic and abstract manner. Rules on meta model level have been defined that capture the complex relationships between the multiple viewpoints. Whenever the modeler changes the value of an attribute, a script is called that checks the current changes against the rule base. Depending on the current change, the algorithm then determines changes that need to be performed to all SOM viewpoints in order to preserve consistency. As the ADOxx platform does not support a repository approach out of the box, such functionality had to be implemented by hand. Moreover, most of the synchronizations would not have been solely realized by a repository. The realization of these synchronization mechanisms has been achieved by adding several attributes to the models and the modeling and relation classes that are not visible to the modeler, but required by the script in the background

Layout Algorithms As denoted at the description of the resource model transformations in section 7.3.2.4, the functionality of the SOM modeling tool has been extended with auto-

layout algorithms for each of the seven realized ADOxx modeltypes. Object Decomposition and Transaction Decomposition Schema provide a tree-based visualization of the decomposition of business objects and business transactions, respectively. The Interaction Schema provides two layout algorithms to the modeler, a *smooth edges* and an *auto-layout* algorithm. The former routes all business transactions in the most possible direct way between the business objects, whereas the latter routes all business transactions either without an edge or only by edges with an angle of 90 degrees. Moreover, if two or more business transactions are related between two business objects, they are automatically shifted up and down in order to prevent overlaps. In the Task-Event Schema, all tasks are vertically grouped by their corresponding business object. Business transactions are only routed vertically, whereas internal events and external events are only routed horizontally.

The Schema of Task Classes provides also a vertical grouping of task specific objecttypes within a certain TAS. Moreover, multiple Schema of Task Classes can be combined in one model. The Schema of Conceptual Classes initially positions all independent objecttypes on the left border of the model, all transaction specific objecttypes are then positioned from left to right according to their behavior specification in the TES. The tool provides an algorithm for positioning all dependent objecttypes either in the center, i.e., *Center COS*, or at the top of the modeling area, i.e., *Align COS*. Finally, the modeler can also iterate over several arrangements of the independent objecttypes on the left border, in order to prevent intersections of relations.

The BPMN models are also initially layouted with the goal of preventing intersections and overlaps. The positioning of the activities is aligned to the positioning of the tasks in the Task-Event Schema.

Notably, all of these layout algorithms aim at increasing efficiency (FUHRMANN AND VON HANXLEDEN, 2010; FUHRMANN, 2011). However, all of them are optional, meaning that the modeler can decide whether a layout algorithm should be applied to a certain view or not. Moreover, initial positioning can be performed by a layout algorithm and then the modeler can change this initial layout to his or her needs. As van Hanxleden already stated, *"[...] given a modeling platform that provides automated drawing capabilities, we can raise the abstraction level of editing activities to work on the structure of the model itself, rather than work on its representation"* (VON HANXLEDEN ET AL., 2012, p. 214).

Model Transformation Algorithms All discussed model transformations, SOM business process models to TAS, COS, and BPMN, are realized following an all-or-nothing principle. Prior to the execution of the transformations, an algorithm checks, whether the

business process model is valid or not. The validation checks, whether the behavior of the process in the Task-Event Schema is completely defined or not as this is a prerequisite for a reasonable transformation execution. If the validation check result is positive, the transformation scripts are executed, otherwise a warning message is shown to the modeler and the transformation is aborted. The resource layer viewpoints are then created as a snapshot of the business process model level visualized at the moment, the transformation has been triggered. Neither are changes on the derived models reflected in the business process model, nor are changes on the business process model propagated to the resource models.

Whenever the business process model changes, the modeler needs to perform the transformation once again in order to adapt the resource model to the new business process. This comes from the fact, that in contrast to the business process model with its several views, the method doesn't define any conceptual foundation to change propagation and consistency management over different levels of the SOM enterprise architecture (see Figure 41). Moreover, most of the refinement operations that can be applied on the resource layer do not have a semantically equivalent operation applicable on the business process layer, e.g., combine BPMN activities into a subprocess, merge objecttypes in the COS, or delete non-automatable task specific objecttypes in the TAS.

7.3.2.6 Realization of the Non-Functional Requirements

Most of the non-functional requirements identified in Table 23 have been captured with the mechanisms and algorithms described in the preceding section. Consequently, the most important non-functional requirements not yet discussed are introduced briefly in the following.

Separate decomposition and refinement A lesson learned from the utilization of formerly developed SOM modeling tools was that the separation of the decomposition and the increase/decrease of the business process level operators increases usability of the tool. This separation allows to aim at two different modeler types: 1) modelers who like to perform a decomposition and immediately after that increase the level of business process model in order to hook up the new business objects and business transactions with the preserved ones; and 2) modelers who are familiar with the SOM method, think more in the decomposition trees compared to the structural/behavioral viewpoints, and therefore like to perform multiple decompositions before starting to increase the business process level.

The SOM tool realized this requirement by strictly separating the decomposition and increase/decrease modeling operations. Moreover, the former can only be triggered in

the decomposition viewpoints whereas the latter can only be triggered in the Interaction Schema and Task-Event Schema.

Zooming Whenever the models get bigger, human beings face the problem of capturing an overview. The multi-view nature of the SOM method further increases this problem. Therefore, it was decided to implement a zoom operator. Zooming allows the modeler to immediately switch between multiple levels of the business process model, i.e., zooming-out from the highest decomposition level back to the initial level and then stepwise zooming-in back to the current, highest decomposition level. As mentioned earlier, zooming has no effect on the relationships between business objects and business transactions. It only changes the currently visualized business process level.

Visualization Except of the layout algorithms introduced earlier, the SOM tools provides further visualization and customization functionality. Simple changes of the font size of the modeling concepts labels or the automatic shortening of labels if they are too long, are implemented. Also, the modeler has the possibility to automatically let the tool adjust the displayed modeling area after an increase/decrease has been performed in TAS and TES. He or she can also customize the quadrant in which a certain SOM viewpoint should be visualized in the simultaneous visualization of the SOM multi-view business process model. Moreover viewpoints can be opened only in the background if the modeler chooses to concentrate e.g., only on the Task-Event Schema for behavioral specification purposes. In the Task-Event Schema, the vertical position of any tasks belonging to a certain business object can be customized. Moreover, the modeler can choose in which models the context information shall be visualized. Furthermore, as the SOM method does not specify the color of the modeling concepts, the modeler can also customize the color of business objects and business transactions.

This is only a selection of tool functionality aiming at customization and usability of the tool.

Validation The SOM tool comes with some rudimentary model validation functionality. With this validation, an algorithm checks, whether the multiple business process views are consistent to each other, e.g., are there any missing business objects or business transactions in a viewpoint or are the attribute values of business objects and business transactions consistent in all viewpoints. Moreover, the validation checks, whether the behavior of the process model is completely specified or not.

A rudimentary HTML visualization of the validation results is presented to the modeler, so he or she is immediately pointed to inconsistencies that need to be resolved manually.

Figure 71 shows the realization of the SOM multi-view modeling tool by reverting to the concepts and components of the ADOxx meta modeling platform. The illustration is only an excerpt, however it highlights the most relevant aspects of the realization.

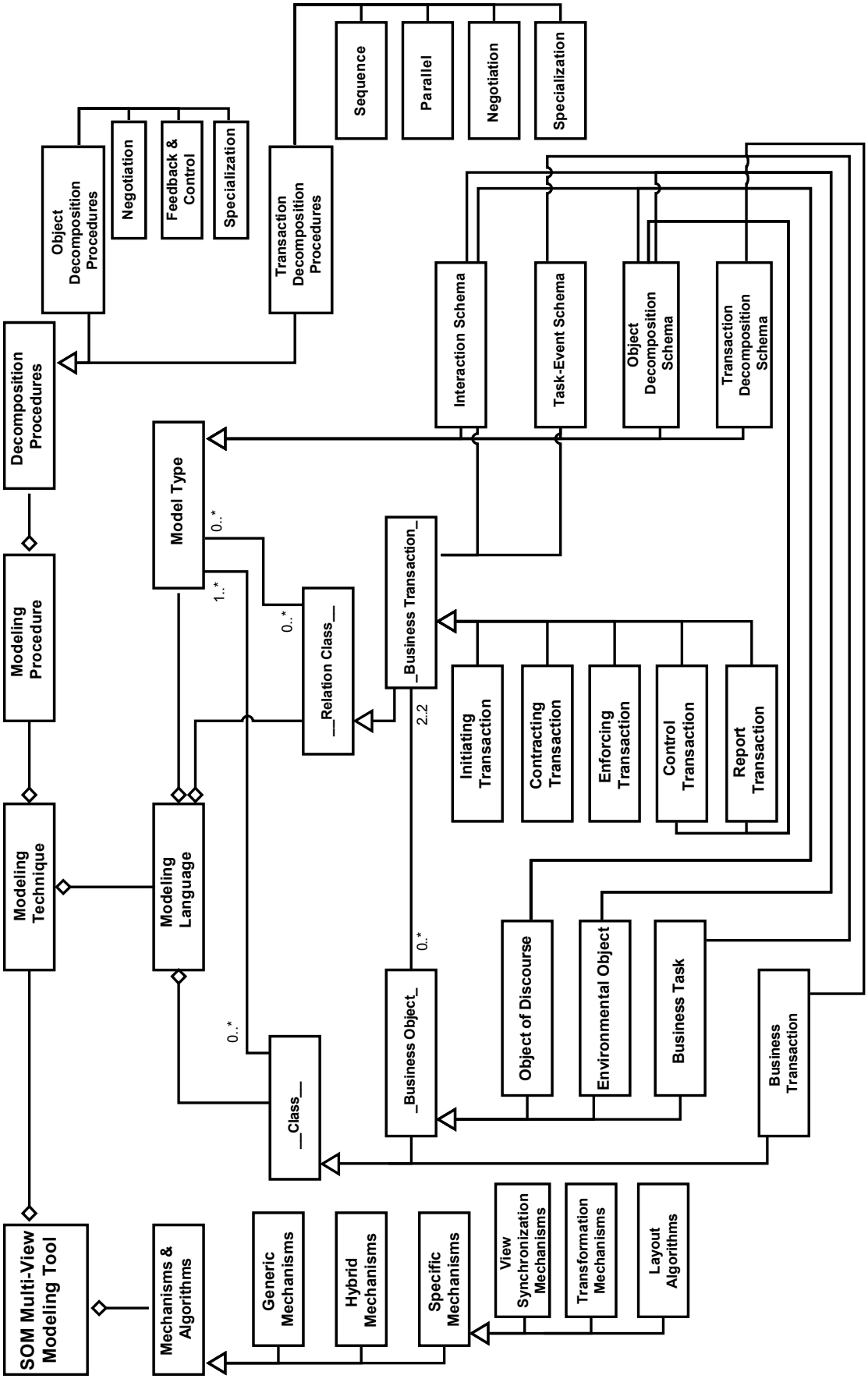


Figure 71: Conceptual Model of the ADOxx SOM business process modeling tool

7.4 Evaluation

Version 3 of the SOM modeling tool has been released in late 2014[40]. The tool is used by experts and students in university courses at the University of Bamberg[41], at the Virtual University of Bavaria[42], and at the University of Vienna[43].

The tool has proven its suitability and utility and therefore, the conceptual design, establishing the requirements the tool has been implemented against, has proven its correctness and completeness. Following the MUVIEMOT approach significantly fostered the creation of a comprehensive requirements specification. Guided through the different steps, the lifecycle guaranteed that no important aspects have been forgotten.

At this point, it should be discussed, whether the SOM modeling method is an appropriate candidate for evaluating the MUVIEMOT approach. SOM is a perfect example for this purpose based on two aspects: First, the SOM method was specifically designed to utilize multi-view modeling. Hence, the semantics of the viewpoints, the modeling languages of the viewpoints and the modeling procedure is comprehensively specified. Therefore, SOM is considered a real multi-view modeling method, not a artificially constructed case in a lab setting. Second, the SOM method incorporates different multi-view modeling principles on the business process modeling layer and the resource layer. Hence, SOM can be considered as two case studies in total as it treats the COS and TAS viewpoints completely different compared to the treatment of the business process model viewpoints.

The specification of **Modeling Scenario** and **Modeling Language** for SOM was straight forward. The SOM method already specifies the modeling language of the viewpoints formally by means of meta models. Moreover, the relationships between the multiple viewpoints and multiple abstraction layers are specified in the enterprise architecture and the SOM procedure model. However, it was interesting, that the specification of the **Modeling Procedure**, i.e., the multi-view modeling use cases, was a very challenging task. Although the SOM method specifies the modeling operations, cf. the SOM decomposition rules in Table 17, in a formal way - what can be considered very rare compared to other commonly applied modeling methods - aligning the operators to the multiple viewpoints and modeling concepts therein revealed inconsistency and incompleteness of the original method specification. For example, rule 2, the decomposition of a business object into two business objects with an optional business transaction routed between the two new business objects. If the modeler decides to decline the optional business transaction, one of the two created business objects might be without any relationship to the other business objects in the business process model - which is by fact against the SOM

[40] Download the SOM modeling tool: `http://www.omilab.org/web/som/download`, last checked: 2015-02-09
[41] University of Bamberg, `http://www.uni-bamberg.de/en`, last checked: 2015-02-09
[42] University of Bavaria, `http://www.vhb.org/en/homepage/`, last checked: 2015-02-09
[43] University of Vienna, `http://www.univie.ac.at/en/`, last checked: 2015-02-09

meta model. Hence, during the design process, it was decided to ask the modeler how many business transactions he or she wants to create during that decomposition, with the requirement of having at least one created.

Another problem that came to surface during the design of the SOM tool was the fact, that modeling operators needed to be applicable in multiple viewpoints. Business transactions are included in three viewpoints, the Transaction Decomposition Schema, the Interaction Schema, and the Task-Event Schema. It was therefore a design decision, pointed to by **MUVIEMOT Modeling Procedure** step, to assign certain modeling operators only to certain viewpoints. Following the example above, business transaction decomposition has only be assigned to the Transaction Decomposition Schema.

The **Viewpoint Dependencies** helped at identifying dependencies that where not clear at first sight. For example, the dependencies between a task in the Task-Event Schema and a business object is only informally specified in the SOM method specification. The same holds for the relationship between a business transaction and a task. In the former case, deleting a business object from the SOM model triggers the deletion of all corresponding tasks, i.e., tasks performed by that business object. In the latter case, changing the name of a business transaction must be reflected by a consistent change of the name of the two corresponding tasks. The MUVIEMOT Viewpoint Dependencies step pointed to these dependencies and therefore fostered a comprehensive and formalized specification of a SOM multi-view modeling tool.

As an obvious criticism to this evaluation it should be noted, that in this case study, the same person who created the conceptual design with MUVIEMOT was also responsible for the development of the modeling tool on the ADOxx meta modeling platform. Therefore, the evaluation can guarantee, that, following the MUVIEMOT approach fosters efficient requirements specification and design of multi-view modeling tools compared to conventional requirements elicitation and tool development from scratch. The goal of MUVIEMOT is furthermore to bridge the gap between a method owner and a tool developer. Hence, future applications of the approach should consider a multi-party setting in order to elaborate on this specific goal.

7.5 Summary

This section described an application of the MUVIEMOT method by means of a comprehensive illustrative scenario. In this regard, section 7.1 first introduced the SOM enterprise modeling method. An emphasis has been on the enterprise architecture and the multi-view nature of the method. On different layers of the enterprise architecture, different multi-view modeling principles are utilized. The actual application of MUVIEMOT was then conducted in section 7.2. The application of all steps of the MUVIEMOT lifecycle has been described thoroughly. Section 7.3 then evaluated the generated conceptual design by a technical implementation based

on the ADOxx meta modeling platform. Hence, the SOM multi-view modeling tool, realized according to the MUVIEMOT conceptual design, was outlined. Finally, section 7.4 evaluated the MUVIEMOT method by highlighting specific benefits identified, and discussing insights gained during the application of the method.

8 The MuVieMoT Modeling Environment

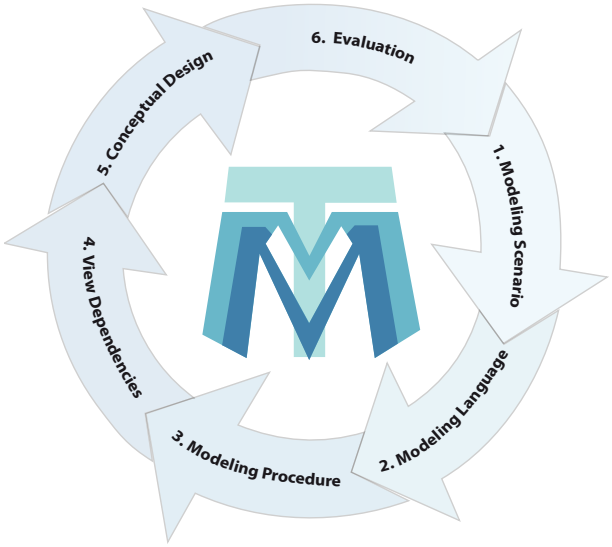


Figure 72: The MuVieMoT modeling environment

This section describes the design and development of a modeling tool for the MuVieMoT approach. The tool aims at providing a more efficient application of the method by means of automatism of manual tasks and further tooling functionality. Hence, the aim is to ease the use of the MuVieMoT method. The tool is realized on the ADOxx meta modeling platform. It can be downloaded free of charge from the project homepage within the Open Models Initiative (OMI)[44].

The tool not only supports method engineers in performing all steps of the MuVieMoT approach by means of conceptual modeling, it also provides the functionality for model-driven development of initial multi-view modeling tools for tool developers. Therefore, the conceptual design, created throughout the modeling process, can be automatically transformed into an ADOxx library, enabling immediate multi-view modeling according to the specification.

This section is structured as follows: Section 8.1 motivates the necessity of the tool and describes the major benefits of it. Functional and non-functional requirements are collected in section 8.2. The developed MuVieMoT modeling environment is then described in section 8.3. The realized model-driven development approach is described in section 8.4. Evaluating the tool by means of mapping the functionality of the developed tool to the specified requirements is performed in section 8.5.

[44] MuVieMoT project page, `www.omilab.org/web/muviemot/`, last checked: 2015-02-20

8.1 Motivation & Aim

With increasing maturity of the MUVIEMOT method and the number of applications of the method, it was identified, that the positive aspects of the method, i.e., guiding the multi-view modeling tool specification process, can be further increased, and the negative aspects, i.e., the informal specification of some of the method's steps and the missing tool support, can be resolved by providing a modeling environment. Atkinson and Kühne define the benefits of a model-driven development approach as follows: *"Instead of requiring developers to use a programming language spelling out how a system is implemented, it allows them to use models to specifying what system functionality is required and what architecture is to be used"* (ATKINSON AND KÜHNE, 2003, p. 36). Hence, a MUVIEMOT modeling tool was designed and developed on the ADOxx meta modeling platform. The specific aims of this modeling tool are as follows:

- Support each step of the MUVIEMOT method with a specifically designed conceptual modeling language.
- Provide model transformations between the models created at different steps of the MUVIEMOT approach.
- Realize the model-driven development of multi-view modeling tools based on the specified conceptual design.
- Increase efficiency by reusing already specified aspects.

In total, the tool has multiple positive effects on applicability and utility of the MUVIEMOT method. *"There is a clear advantage in providing this kind of automated support - namely, that there is a single specification written by the designer, which is used to both verify and implement the system, thus avoiding the significant problem of the introduction of differences between the verified and implemented versions of the system"* (WOOD ET AL., 2008, p. 1357).

The focus of this tool is on the usage of the design specification as a basis for model-driven development. Voelter et al. criticize the isolated elicitation of requirements and not considering them during the implementation phase. *"Requirements often play second fiddle in software development projects. The tools for managing requirements often just support "numbered lists of prose paragraphs", and they don't integrate well with the tools used for implementing the system"* (VOELTER ET AL., 2013, p. 1). The focus of the MUVIEMOT tool is therefore on using the requirements as initiator of a model-driven development process. By providing specifically designed conceptual modeling languages for each step, rudimentary validity of the models is also guaranteed.

8.2 Requirements

In the following, the requirements of a modeling tool supporting the development of multi-view modeling tools are outlined. The requirements specification is separated into three parts: First, section 8.2.1 describes requirements originating from the MUVIEMOT modeling method, referred to as MUVIEMOT *requirements*. Second, requirements with an emphasis on the codification of the specific characteristics of the multi-view modeling domain are captured as *application domain requirements* in section 8.2.2. Third, *non-functional requirements* are discussed in section 8.2.3.

On an accumulated level, the general requirement for such a tool is to support method engineers efficiently with concepts, models, and functionality, in the process of specifying the conceptual design of multi-view modeling tools. Wherever possible, the requirements are kept on a generic level. For some requirements however, e.g., the model processing requirements, a dependency to the tool development platform and/or the platform the specified modeling tool should be implemented on, is inevitable. In such cases, requirements are specifically defined for the ADOxx meta modeling platform.

8.2.1 MUVIEMOT Requirements

The following requirements originate from the aim of supporting method engineers in applying the MUVIEMOT method.

MVMT-1: MUVIEMOT Syntax A modeling tool supporting the MUVIEMOT approach shall include the modeling languages' syntax. As the MUVIEMOT method proposes multiple steps in its lifecycle, each step shall be supported by dedicated syntax specification, presented to the users of the tool as ADOxx modeltype. Hence, the tool shall support a modeltype for each of the following MUVIEMOT steps: *Modeling Scenario*, *Modeling Language*, *Modeling Procedure*, *Viewpoint Dependencies*, and *Conceptual Design*. The evaluation step has no corresponding model as it is considered a non-automatable, cognitive step that uses a created conceptual design (or even a implemented prototype of it) and compares it to the modeling method it is designed for, e.g., by means of case studies or expert interviews.

MVMT-2: MUVIEMOT Notation In order to realize the benefits of graphical modeling languages and modeling tools, it is necessary to map the abstract conceptual specification of the MUVIEMOT modeling concepts to *concrete graphical visualizations*. Consequently, each concept of each MUVIEMOT step shall have an *intuitive interpretable notation*.

MVMT-3: MUVIEMOT Semantics The semantics of each concept of each MUVIEMOT step shall be specified by attributes, presented to the tool users by means of ADOxx notebooks.

The tool shall limit the edit operations to a semantically reasonable set of valid changes.

MVMT-4: MUVIEMOT Modeling Procedure The modeling procedure, defining steps and results that are performed by the modeler in the act of applying the MUVIEMOT method, shall be supported in the modeling tool in an appropriate manner. Therefore, the tool shall guide the modeler during the application of the method. Where appropriate, automatic transformations between MUVIEMOT steps, respectively their corresponding models, shall be supported.

MVMT-5: Consistency The MUVIEMOT method itself can be considered as a multi-view modeling method. It is therefore important to treat the multiple ADOxx modeltypes as viewpoints on one overarching conceptual design specification. Hence, consistency mechanisms shall be established, enabling the consistency preserving generation of a conceptual design with the tool.

8.2.2 Application Domain Requirements

Multi-view modeling methods, as introduced in section 5.1, are specialized modeling methods enabling the usage of several, interrelated modeling viewpoints in order to describe a complex system comprehensively. The specialized application domain and the specific characteristics of such methods demand for specific requirements on a supporting modeling tool. Moreover, requirements enabling the generated models to not solely serve as knowledge bases for specification, documentation, or communication purposes, but also for further model processing purposes, are defined.

APPD-1: Viewpoint Specification The tool shall enable a method engineer to specify a modeling viewpoint intuitively. The specification consists of the concepts considered by the viewpoint, the viewpoint derivation approach (i.e., utilizing a projective or selective relationship to one or more meta models), and attributes, describing the viewpoint (e.g., a name for the viewpoint, stakeholders addressed by the viewpoint, concerns addressed by the viewpoint).

APPD-2: Multi-View Modeling Use Case Specification The specific characteristics of multi-view modeling require a thorough consideration of modeling operations performed by the modeler. Therefore, modeling operations shall be aligned to a viewpoint or a set of viewpoints, thereby highlighting their scope.

APPD-3: Viewpoint Relationship Specification By analyzing the specified viewpoints and their derivation from the meta models, a first set of relationships between the viewpoints should be automatically processed and generated by the tool. However, this is

not satisfying. The tool shall allow for efficient specification of additional, non-trivial relationships between viewpoints. The relationships shall be drilled down to modeling concepts and their attributes.

APPD-4: Viewpoint Transformation Specification The tool shall enable the method engineer to define transformations between viewpoints using a model-driven approach. Different transformation types, e.g., bidirectional or directional shall be definable on a suitable abstraction level.

APPD-5: Visualization of View Dependency In order to provide the method engineer and the tool developer with an abstract view on the multi-view modeling method, precisely, on the viewpoints and the relationships they adhere, a modeling tool shall enable the generation of a viewpoint dependency diagram. This diagram shall spotlight the dependencies between the viewpoints, the meta models, the attributes and the transformations in an intuitive and human understandable way.

APPD-6: Tool Generation Following the goal of rapid implementation of multi-view modeling tools, the MUVIEMOT modeling tool shall enable the transformation of the specified models into initial implementations of modeling tools based on the ADOxx meta modeling platform. This requirements enables two benefits: (1) model-driven development of initial multi-view modeling tools, and (2) increase of efficiency by replacing knowledge-intensive and technical conceptualization steps of the platform with more intuitive modeling and model editing operations realized with the tool, e.g., the specification of the meta model using a modeling approach instead of the tree-based hierarchy structure the ADOxx platform provides.

APPD-7: Generation of Consistency Mechanisms The specification of dependencies in the Viewpoint Dependencies model shall be transformed into machine-processable rule bases or synchronization mechanisms. Depending on the development platform used, AdoScript, Object Constraint Language (OCL) constraints, or Query View Transformation (QVT) rules are appropriate notations.

APPD-8: Multiple Visualization A common use case of multi-view modeling methods is that a certain concept is considered in multiple viewpoints. It is therefore beneficial, if the modeling tool would provide the possibility to define a viewpoint-dependent visualization.

8.2.3 Non-Functional Requirements

From a non-functional perspective, the MUVIEMOT tool shall follow its aim and foster the specification of conceptual designs of multi-view modeling tools. Due to the complex application domain, increasing the usability of the tool for modelers shall be of high importance. Moreover, classical non-functional requirements like usability, scalability, and robustness shall be also considered.

NFR-1: Application Scenarios In order to increase the acceptance of the modeling tool and to account for the different types of modelers, it shall be able to use the modeling tool in different ways. The first, and recommended, way of utilizing the tool shall be strictly aligned to the procedural approach of the MUVIEMOT method. Consequently, all models are created accordingly to the MUVIEMOT life cycle. However, it would be helpful if experienced modelers are able to create the different models on their own regard i.e., in a non-anticipated sequence.

NFR-2: Modeling Guidance As the MUVIEMOT modeling tool is very complex due to the different models and steps it comprises, providing guidance to the modeler wherever possible is of major importance. As a consequence, the tool shall be made publicly available, together with a documentation, case studies, tutorials and videos, showcasing the intended usage of the tool.

NFR-3: Help and Tooltips For each attribute shown to the modeler, tooltips shall be provided, allowing the immediate access to a description of the semantics and purpose of the attribute.

NFR-4: Backtracking If changes were performed to models of advanced MUVIEMOT steps, update mechanisms shall be provided between dependent preceding steps of the procedure. This requirement is related to *MVMT-5: Consistency* requirement.

8.3 The MUVIEMOT Modeling Tool

The MUVIEMOT modeling tool has been developed on the ADOxx meta modeling platform (see section 7.3.1 for an introduction to ADOxx). The development took place as a research project, participating in the Open Models Initiative Laboratory (OMiLAB)[45]. The tool can be downloaded free of charge, including all sources necessary to customize or extend the tool's functionality.

[45] The MUVIEMOT project page, `http://www.omilab.org/web/muviemot/` last checked: 2015-02-19

The following sections describe the multiple ADOxx modeltypes that have been realized to capture the needs of the different steps of the MUVIEMOT method. For each modeltype introduced, a conceptual model of the ADOxx meta model, a tabular specification of the implemented notation, and a sample model, created using the same use case as in the previous section, the SOM method, is described. The different MUVIEMOT steps are not discussed thoroughly, as they have been introduced already in section 6.3.

8.3.1 Modeling Scenario

The conceptual model for the MUVIEMOT **Modeling Scenario** is illustrated in Figure 73. Real world and sub-area of the real world are not considered in the conceptual model. These aspects are defined with the goals and the metaphor to be pursued during the modeling process. Multiple goals can be specified with the tool. Moreover, models and views, being instances of meta models and viewpoints, respectively, are not considered to be included in the modeling scenario.

A modeler has one to many goals in mind. He or she is guided by one to many metaphors, one metaphor for each meta model of the modeling scenario. The modeler interacts with at least two viewpoints. Each viewpoint is related to one or more meta models. In this regard, the modeling language is specified by a meta model, whereas viewpoints reuse concepts that are introduced in one or more meta models.

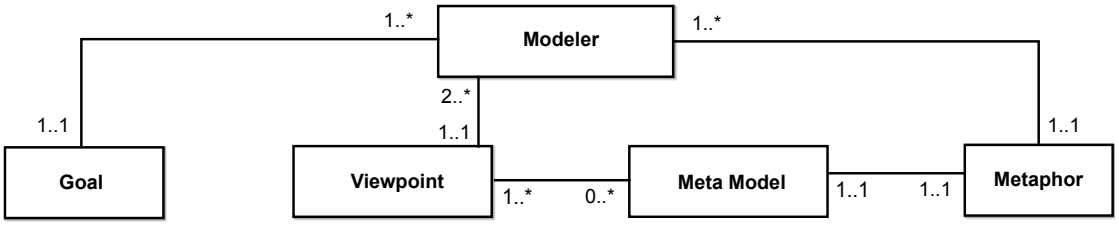
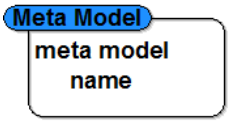


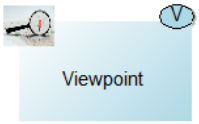
Figure 73: Conceptual model of the *Modeling Scenario* ADOxx modeltype

Table 26 illustrates the implemented notation of the concepts utilized in the modeling scenario ADOxx modeltype. The **Modeler** is realized by a iconic stick figure, the **Goal** by a golden pyramid, the **Metaphor** by a blue star, the **Meta Model** by a rectangle with a blue label on the upper left border, and the **Viewpoint** by a blue rectangle with a loupe on the upper left and a 'V' in an ellipse on the upper right border.

Three relationship types are supported in the modeling scenario modeltype: A **related-to** relationship, visualized by a directed black arrow; a **projection** relationship, visualized by an undirected black line with the label "Projection" centered on the line; and a **selection** relationship, also visualized by an undirected black line but this time with the label "Selection".

Table 26: Notation and semantics of the *Modeling Scenario* modeltype

NOTATION	SEMANTICS
ADOxx Modeling Classes	
	MODELER The modeler is in the center of the modeling scenario. The multi-view modeling tool is to be specifically designed for the purpose of helping human beings in creating multi-view models of a system. The MUVIEMOT tool enables the definition of multiple modelers, i.e., with different quality levels or fulfilling different roles.
	GOAL With this element, the tool user is enabled to specify goals that should be pursued by the modeler while creating the models. Goals might originate from multiple stakeholders or several application scenarios, the created models should be used for, e.g., specification, analysis, or design. These goals also contribute in delimiting the relevant aspects of the real world to be captured in the models. GOALS: a table of goals, specified by the attributes: *short name, description, stakeholder, and importance*.
	METAPHOR The metaphor delimits the perception of the real world by the modeler while creating models. Together with the goals, the metaphor delimits the relevant aspects to be considered during the model creation process. DESCRIPTION: a textual description of the metaphor.
	META MODEL Using this concepts, MUVIEMOT users are enabled to informally define meta models of the multi-view modeling method. The user only has to specify the name of the meta model in this step, later he or she can create a different model, thereby specifying all constituents of the meta model. This meta model can then be referenced in the modeling scenario (see next attribute above).

	RELATED META MODEL: an ADOxx *Interref* (intermodel reference), pointing to the ADOxx model of type *Meta Model* that specifies the meta model with all its concepts and relationships.
	VIEWPOINT This concept can be used to define viewpoints of the multi-view modeling method informally. Specification of the viewpoint's constituents is performed in a later step and in a different ADOxx model. When created, this model can be reference to in the modeling scenario. RELATED VIEWPOINT MODEL: an ADOxx *Interref* (intermodel reference), pointing to the ADOxx model of type *Viewpoint Model* that specifies the Viewpoint with all its concepts and relationships. GENERATE VIEWPOINT MODEL: a program call the user can use to create an empty Viewpoint Model he or she can subsequently use to specify the viewpoint's constituents. DERIVED FROM VIEWPOINT: specification of an extension relationship between viewpoints by means of an ADOxx *Interref*. All modeling classes and relation classes of the extended viewpoint are included in the extending viewpoint. NOTEBOOK SPECIFICATION: specification of the attribute representation (the ADOxx Notebook) of a viewpoint. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the Notebooks. GRAPHICAL REPRESENTATION: specification of the background modeling area visualization of a viewpoint; by default, a white drawing area will be shown. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the method representation.

ADOxx Relation Classes

<Description>	RELATED-TO This, very general relationship can be used to create a relation between the modeler and instances of goal, metaphor, viewpoint, and meta model. While connecting a modeler instance with one of the aforementioned concepts, the tool fires an event and executes AdoScript code. This code determines the ingoing and outgoing concepts and sets the label of the relation correspondingly, i.e., if the user relates a modeler with a goal, the tool automatically visualizes the label *pursues*. DESCRIPTION: the semantics of the relationship, visualized as a label. ENDPOINTS: connecting *modeler* instances with *goal, metaphor, viewpoint, and meta model* instances.
Projection	PROJECTION This relationship type can be used to relate viewpoint instances to meta model instances by means of a *projection operator* applied to the meta model. Using this relationship type indicates, that the syntax of a viewpoint is composed of several modeling concepts of one meta model. ENDPOINTS: connecting *viewpoint* instances with *meta model* instances.
Selection	SELECTION This relationship type can be used to relate viewpoint instances to meta model instances by means of a *selection operator* applied to the meta model. Using this relationship type indicates, that the syntax of a viewpoint is composed of modeling concepts of different meta models. ENDPOINTS: connecting *viewpoint* instances with *meta model* instances.

Figure 74 illustrates the modeling scenario for the business process modeling layer of the SOM method.

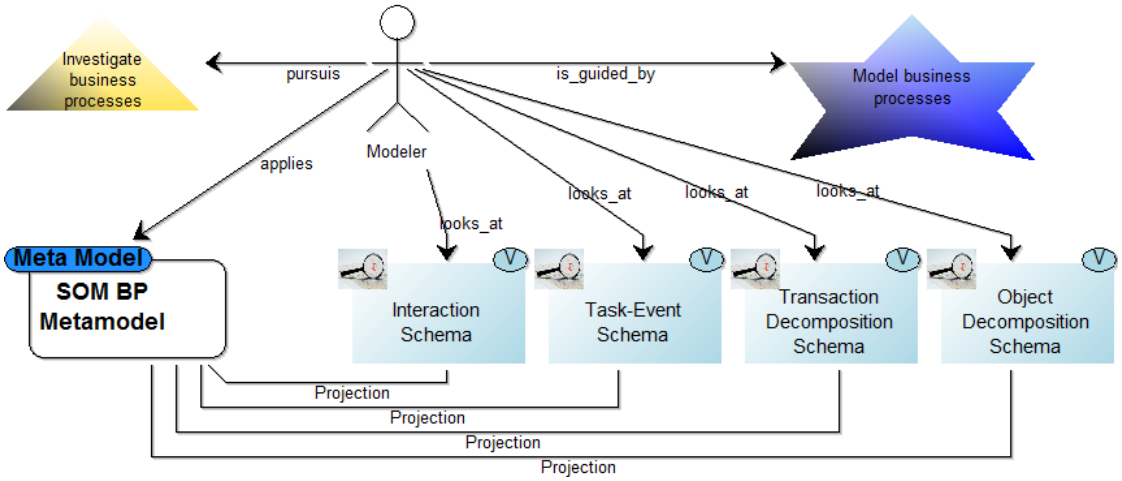
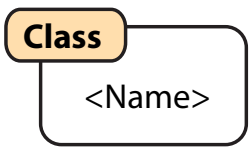


Figure 74: SOM Modeling Scenario model created with the MUVIEMOT tool

8.3.2 Modeling Language

The specification of the modeling language utilized by the multiple viewpoints plays a mature role in the specification of a modeling tool. The MUVIEMOT modeling tool therefore supports two different modeltypes for specifying the meta model and the viewpoint. The assumption is that no two versions of a meta model exist, however, one meta model may be utilized differently in two viewpoints. Moreover, the meta model model is supposed to be platform- and implementation-independent, whereas the viewpoint model can be considered a realization of a meta model with a certain meta modeling or tool development platform. Hence, although both modeltypes have the same concepts, their semantics and usage scenarios should be distinguished. Table 27 provides an overview of the provided concepts for both, the *meta model* modeltype and the *viewpoint model* modeltype.

Table 27: Notation and semantics of the *Meta Model* and *Viewpoint Model* modeltypes

NOTATION	SEMANTICS
ADOxx Modeling Classes	
	MODELING CLASS In analogy to the UML definition of a class, a modeling class describes "a set of objects that share the same specifications of features, constraints, and semantics." (OBJECT MANAGEMENT GROUP (OMG), 2011d, p. 48).

NAME: unique name of the modeling class.

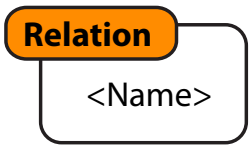
ATTRIBUTES: table of attributes[46] defined by: *AttrName, Type, Default Value, Attribute Information, and MultiLineString*.

ISABSTRACTCLASS: defines, whether the modeling class is abstract or not.

ISDERIVEDFROMSUPERCLASS: a reference to a modeling class the current one is inherited of. All attributes of the referenced class will be inherited by this class.

GRAPHICAL REPRESENTATION: specification of the notation for the modeling class. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the notation.

NOTEBOOK SPECIFICATION: specification of the attribute representation of the modeling class to the modeler. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the Notebook.



RELATION CLASS

The concept of relation class is used to specify and constrain the valid relationships between modeling classes. A relation class needs to specify the modeling classes that can be connected by it, ingoing and outgoing.

NAME: unique name of the relation class.

FROM MODELING OBJECT: reference to a modeling class of the same viewpoint model, instances of this relation class can be outgoing from.

TO MODELING OBJECT: reference to a modeling class of the same viewpoint model, instances of this relation class can be ingoing to.

ATTRIBUTES: table of attributes defined by: *AttrName, Type, Default Value, Attribute Information, and MultiLineString*.

[46] the attributes are specific for the ADOxx meta modeling platform.

GRAPHICAL REPRESENTATION: specification of the notation of the relation class. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the notation.

NOTEBOOK SPECIFICATION: specification of the attribute representation of the relation class to the modeler. The syntax of the ADOxx platform needs to be utilized in order to enable model-driven development functionality also for the Notebook.

As introduced earlier, central concepts of ADOxx meta models are *Modeling Classes*, *Relation Classes* and relationships between them. therefore, the Viewpoint Model has also two modeling concepts: **Modeling Class**, visualized by rectangle with the label 'Class' on the upper left border; and **Relation Classes**, also visualized by a rectangle, but with the label 'Relation' on the top left border. Modeling Classes and Relation Classes can be connected with each other by means of **association** or **generalization** relationships. Both can be configured in the ADOxx Notebook of the relation classes and modeling classes in the meta model models and viewpoint models. Notably, the ADOxx platform does not provide inheritance of relation classes. Therefore, the MUVIEMOT modeling tool provides the *isDerivedFromSuperclass* attribute only to modeling classes.

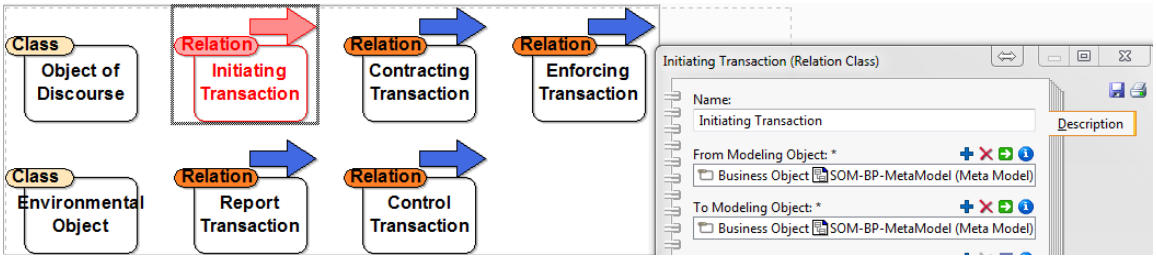


Figure 75: SOM Interaction Schema Viewpoint Model created with the MUVIEMOT tool

Figure 75 shows the realization of the SOM Interaction Schema by means of a MUVIEMOT viewpoint model. The modeling classes *Object of Discourse* and *Environmental Object* are accompanied with the relation classes *Initiating Transaction, Contracting Transaction, Enforcing Transaction, Report Transaction, and Control transaction*. Exemplary for the specification of relation classes, the ADOxx Notebook of the Initiating Transaction is visualized on the right side. It illustrates, that Initiating Transactions can relate *Business Objects* with *Business Objects*. Notably, the class business object is specified as an abstract class in the SOM business process meta model. *Object of Discourse* and *Environmental Object* are subclasses of business

object. Consequently, through this inheritance relationship, both can be related by Initiating Transactions as outgoing and ingoing node.

8.3.3 Modeling Procedure

The Modeling Procedure step of MUVIEMOT combines the viewpoints with the behavioral aspects of modeling by means of *Multi-View Modeling Use Cases*. Hence, central elements of the ADOxx modeltype for the Modeling Procedure specification are **Viewpoint** and **Use Case**. Viewpoints are further distinguished into *Triggered-In Viewpoints* and *Effect-On Viewpoints*. Consequently, use cases can be related to Triggered-In Viewpoints by means of **triggers** relationships and to Effect-On Viewpoints by *effects, and conditionally_effects* relationships.

Figure 76 visualizes the constituents of the Modeling Procedure ADOxx modeltype conceptually. The name of abstract concepts is written using a bold font.

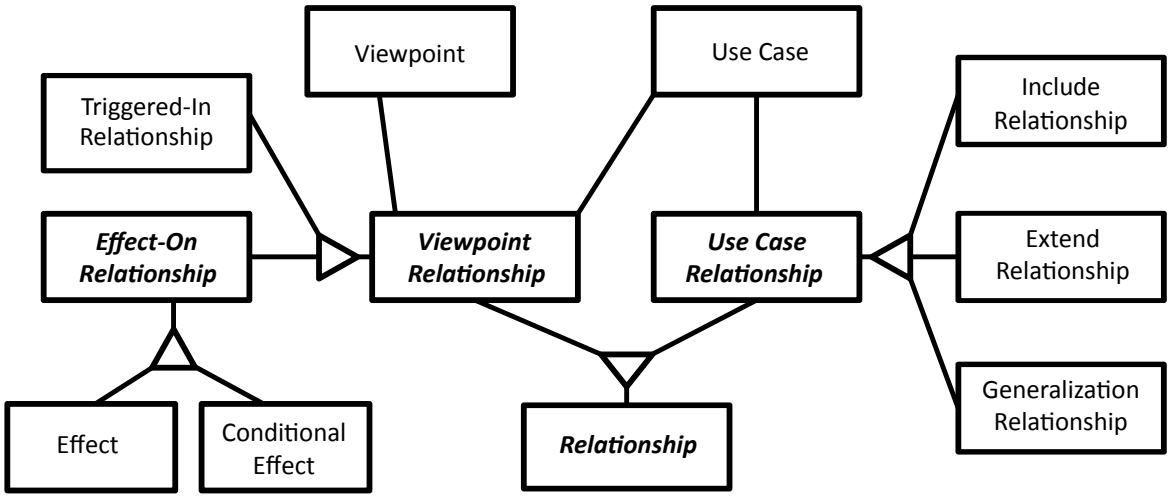
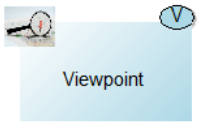
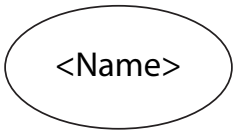


Figure 76: Conceptual model of the *Modeling Procedure* ADOxx modeltype

Table 28 summarizes the constituents of the Modeling Procedure ADOxx modeltype. *Viewpoints* are represented as rectangles with a loupe and the label 'V' (like in the viewpoint model), *Triggered-In Relationships* as continuous directed arrows, *Effect Relationships* as directed double-lined arrows, and *Conditional Effect Relationships* as directed dashed arrows. The notation of the adopted concepts stays the same as for conventional UML use case diagrams (i.e., use cases are visualized as ellipses, and the Use Case Relationships like include, extend and generalization as defined in the standard (OBJECT MANAGEMENT GROUP (OMG), 2011d)).

Table 28: Notation and semantics of the *Modeling Procedure* modeltype

NOTATION	SEMANTICS
ADOxx Modeling Classes	
	VIEWPOINT This concept can be used to define viewpoints of the multi-view modeling method informally. Specification of the viewpoint's constituents is performed in a different step and in a different ADOxx model. ATTRIBUTES: All attributes as specified for the viewpoint modeling class in Table 26.
	USE CASE In analogy to the UML definition of use cases, this concept is used to describe a certain system behavior by means of a functionality of the modeling tool triggered by the modeler. NAME: unique identifier for the use case.
ADOxx Relation Classes	
	TRIGGERED-IN The triggered-in relationship type is used to specify, that a certain use case can be triggered in a certain viewpoint. ENDPOINTS: from a *triggered-in viewpoint* to an *use case*.
	CONDITIONAL EFFECT This relationship type is used to specify a conditional effect on a certain viewpoint, resulting from the execution of a certain use case. ENDPOINTS: from an *use case* to an *effect-on viewpoint*.
	EFFECT This relationship type is used to specify a direct effect on a certain viewpoint, resulting from the execution of a certain use case.

	ENDPOINTS: from an *use case* to an *effect-on viewpoint*.
<<extends>>	EXTEND This relation enables the extension of one use case by another use case, consistent to the specification of the UML extend relationship type. ENDPOINTS: from an extending *use case* to an extended *use case*.
<<includes>>	INCLUDE This type of relation is used to model the inclusion of the modeling actions defined in one use case by another use case, consistent to the specification of the UML include relationship type. ENDPOINTS: from an including *use case* to an included *use case*.

Figure 77 shows an excerpt of the SOM business process *modeling procedure* model created with the MUVIEMOT tool. When creating a modeling procedure model for the first time, the tool automatically generates all viewpoints based on the viewpoint models found in the current ADOxx model group. Moreover, the tool provides two areas, the viewpoints are automatically positioned in: a *Triggered-In* and an *Effect-On* area.

For each identified viewpoint model, a viewpoint element is positioned in the *Triggered-In* area located at the left margin of the modeling canvas, and another viewpoint element positioned in the *Effect-On* area located at the right margin of the modeling canvas. For readability reasons, the name of the viewpoints is also directly imported and adopted for the two created elements, hereby appending the original name of the viewpoint with a '[T]' or '[E]' suffix. This initial visualization should foster intuitive understanding of the modeling procedure step of MUVIEMOT. Notably, this initial positioning can be adopted by the modeler.

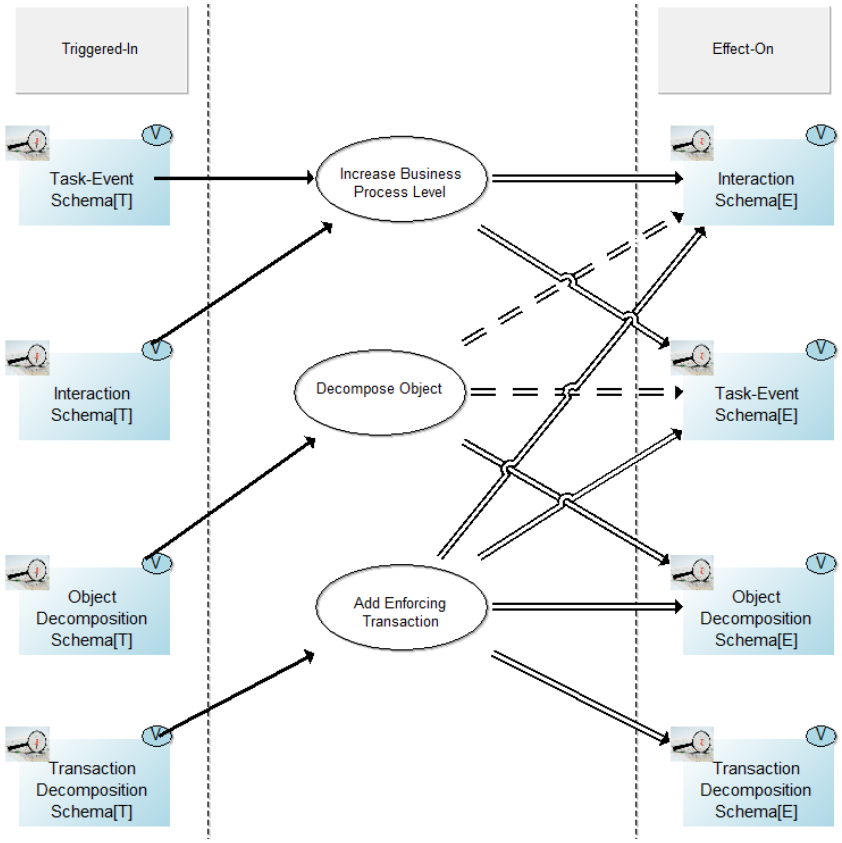


Figure 77: Excerpt of the SOM Modeling Procedure model created with the MUVIEMOT tool

8.3.4 Viewpoint Dependencies

The ADOxx Viewpoint Dependency modeltype is conceptually visualized in Figure 78. Abstract concepts are written in bold italics. Hence, instances of the Viewpoint Dependency modeltype comprise *is-used-in* relationships that connect exactly one *viewpoint* with one *modeling concept*. In this step, the user abstracts from modeling and relation classes. Syntactic dependencies are automatically detected by the tool by traversing all Viewpoint Model models in the current ADOxx model group.

The algorithm creates an inverted list with a modeling concept, independently if it is a modeling class or a relation class, as key and a list of all viewpoints it is used in as value. Hence, the name of a modeling concept needs to be unique in the multi-view modeling method for the algorithm to work properly. Semantic dependencies may then be added manually by the MUVIEMOT user. By looking at the ADOxx Notebook of a modeling concept, the modeler is presented a tabular visualization of all dependencies. He or she can edit this table and add additional syntactic and semantic dependencies.

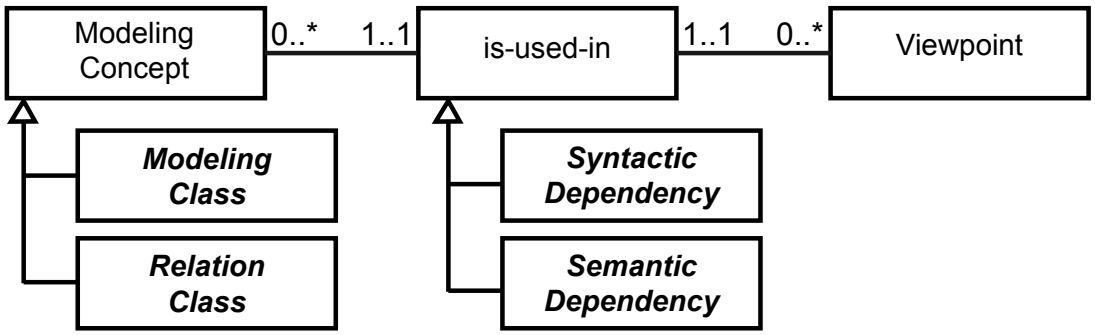
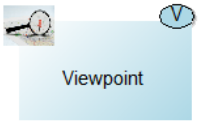


Figure 78: Conceptual model of the *Viewpoint Dependency* ADOxx modeltype

Table 29 summarizes the constituents of the Viewpoint Dependencies ADOxx modeltype. *Viewpoints* are represented as rectangles with a loupe and the label 'V' (like in the Viewpoint Model), *Modeling Concepts* as blue hexagons, and *is-used-in* relationships as dotted black lines with the label 'is-used-in' centered on top of the relation.

Table 29: Notation and semantics of the *Viewpoint Dependencies* modeltype

NOTATION	SEMANTICS
ADOxx Modeling Classes	
	VIEWPOINT This concept can be used to define viewpoints of the multi-view modeling method informally. Specification of the viewpoint's constituents is performed in a different step and in a different ADOxx model. ATTRIBUTES: All attributes as specified for the viewpoint modeling class in Table 26.
	MODELING CONCEPT A modeling concept is the abstract representation of modeling classes and relation classes. This enables concepts, being represented in one viewpoint as modeling class to be mapped to the same concept that is represented by a relation class in a different viewpoint. Consequently, the name of the modeling concepts needs to be unique for the complete multi-view method.

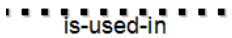
	NAME: identifier for the modeling concept, derived from the Viewpoint Model or manually edited by the user. VIEWPOINT DEPENDENCIES: table of dependencies, each specified by the following set of attributes: *Id, frompart (Concept or Attribute), fromattribute, frommodel, cardinality, topart (Concept or Attribute), toattribute, and tomodel*. This attribute is used for specifying the consistency dependencies in more detail. It distinguishes between 1:1 dependencies and very fine-grained attribute-dependencies.
ADOxx Relation Classes	
	IS-USED-IN Relates modeling concepts with viewpoints they are used in. ENDPOINTS: from a *modeling concept* to a *viewpoint*.

Figure 79 illustrates an excerpt of the generated Viewpoint Dependencies model for SOM business process modeling. The figure shows, that the concept *Task* is only used in the *Task-Event Schema*, whereas *Initiating Transaction*, *Object of Discourse* and *Business Transaction* are used in two Viewpoints. Hence, the concept Initiating Transaction for example, is used in the Viewpoints Task-Event Schema and Interaction Schema.

At the bottom of Figure 79, the ADOxx Notebook of the Initiating Transaction is visualized. It shows the automatically added entries to the View dependencies table. For each syntactic dependency identified, MUVIEMOT creates the is-used-in relationship and adds an entry to the table. The entry specifies that the whole Initiating Transaction concept is to be synchronized between the Interaction Schema and the Task-Event Schema Viewpoints.

These consistency dependencies, specified on viewpoint (i.e., meta model) level, enable their application for any instance of the viewpoints created. Moreover, the specified dependencies are considered during the model-driven development process (see section 8.4). For each dependency a synchronization script is created that ensures consistency between the multiple viewpoints.

8.3.5 Conceptual Design

Referring to the conceptual model of the Conceptual Design step of MUVIEMOT, illustrated in Figure 38, the only concepts considered in a Conceptual Design ADOxx modeltype are **Functional Requirements**, **Non-Functional Requirement**, and **Consistency Requirements**. The consistency requirements are captured as part of the functional requirements specification. In

the current version of the MUVIEMOT tool, no relationships exists between requirements elements. In future versions, however, it might be worth investigating whether it is beneficial to realize semantic relationships, e.g., like those utilized in feature models or requirements models.

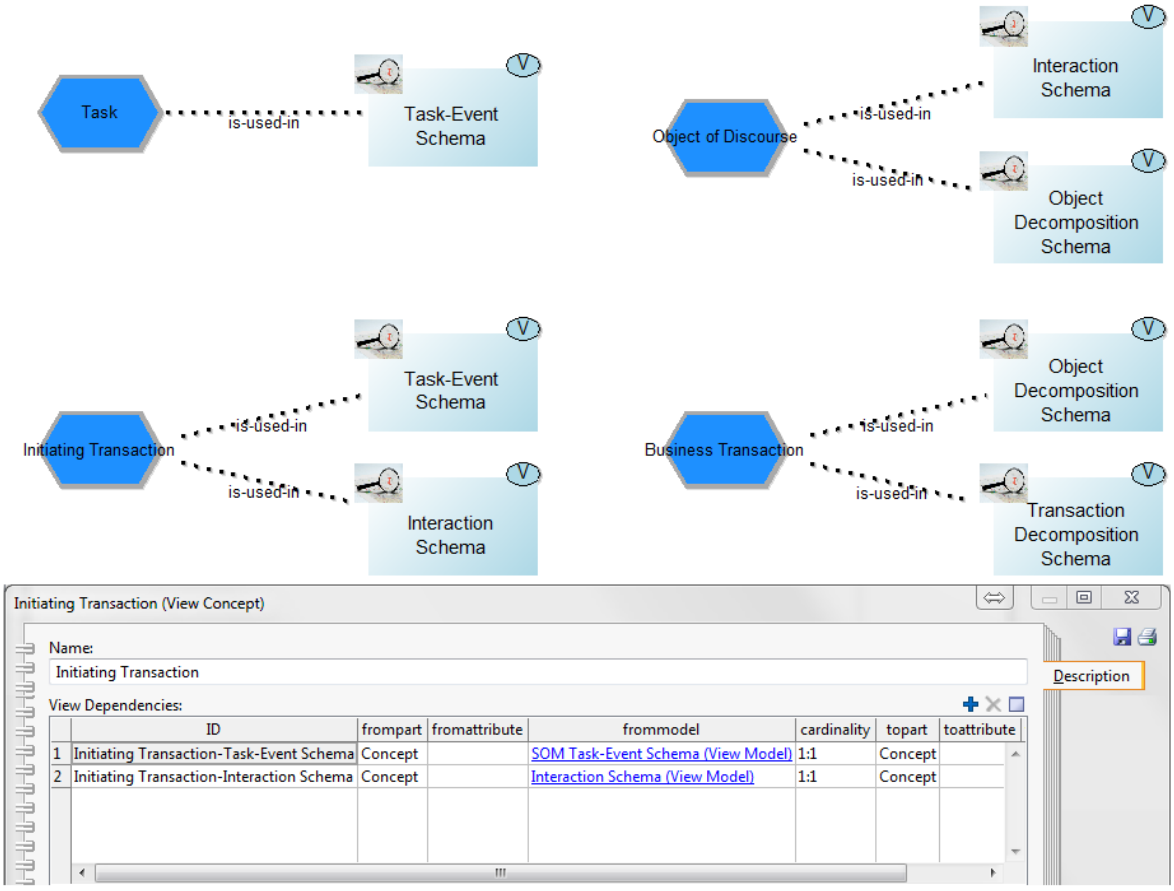


Figure 79: Excerpt of the SOM Viewpoint Dependencies model created with MUVIEMOT

Table 30 summarizes the constituents of the *Conceptual Design* ADOxx modeltype: *Functional Requirements* are visualized as rectangles with the label 'FR' on blue background at the upper left corner, *Non-Functional Requirements* are also visualized as rectangles but with the label 'NFR' on green background in the upper left corner. For both concepts, the name of the concept is visualized as a label in the center of the rectangle. Hence, if the modeler changes the name of the requirement, this change is immediately reflected in the visualization.

The actual specification of the requirements by means of the attributes introduced in section 6.3.5 takes place within the ADOxx Notebook of the requirement. A tabular representation comprising all criteria introduced earlier is presented to the modeler. In this regard, efficient specification of the requirements is provided.

Table 30: Notation and semantics of the *Conceptual Design* modeltype

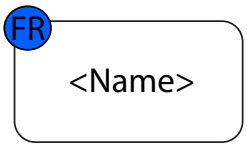
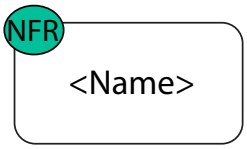
NOTATION	SEMANTICS
	FUNCTIONAL REQUIREMENT NAME: identifier of the functional requirement, visualized in the center of the rectangle. REQUIREMENTS SPECIFICATION: a table of requirements comprised by the functional requirement. A requirement is specified by the following attributes: *Object, Operator, Function, Effect, and Consistency* (cf. section 6.3.5).
	NON-FUNCTIONAL REQUIREMENT NAME: identifier of the non-functional requirement, visualized in the center of the rectangle. REQUIREMENTS SPECIFICATION: a table of requirements clustered by the non-functional requirement. A requirement is specified by the following attributes: *Object, Operator, Function, Effect, and Consistency* (cf. section 6.3.5).

Figure 80 illustrates an excerpt of the Conceptual Design model for SOM business process modeling, created with the MUVIEMOT tool. Moreover, on the bottom of Figure 80, the ADOxx Notebook of the functional requirement *Decomposition* shows the two comprised requirements, i.e., *Decompose Object* and *Decompose Transaction*.

These requirements have been automatically generated by the MUVIEMOT tool. When creating a Conceptual Design model for the first time, the tool traverses through all *Modeling Procedure* models located in the current ADOxx model group and transforms each identified use case into a functional requirement, using the name of the use case also for the requirements name.

Notably, the graphical visualization of the requirements is not perfect, yet. However, looking at the relevant literature on requirements modeling, a common and established standard is lacking. Moreover, all of the found visualizations are equal or less intuitive than the visualization currently implemented in MUVIEMOT. Nevertheless, future research will focus on improving the notation.

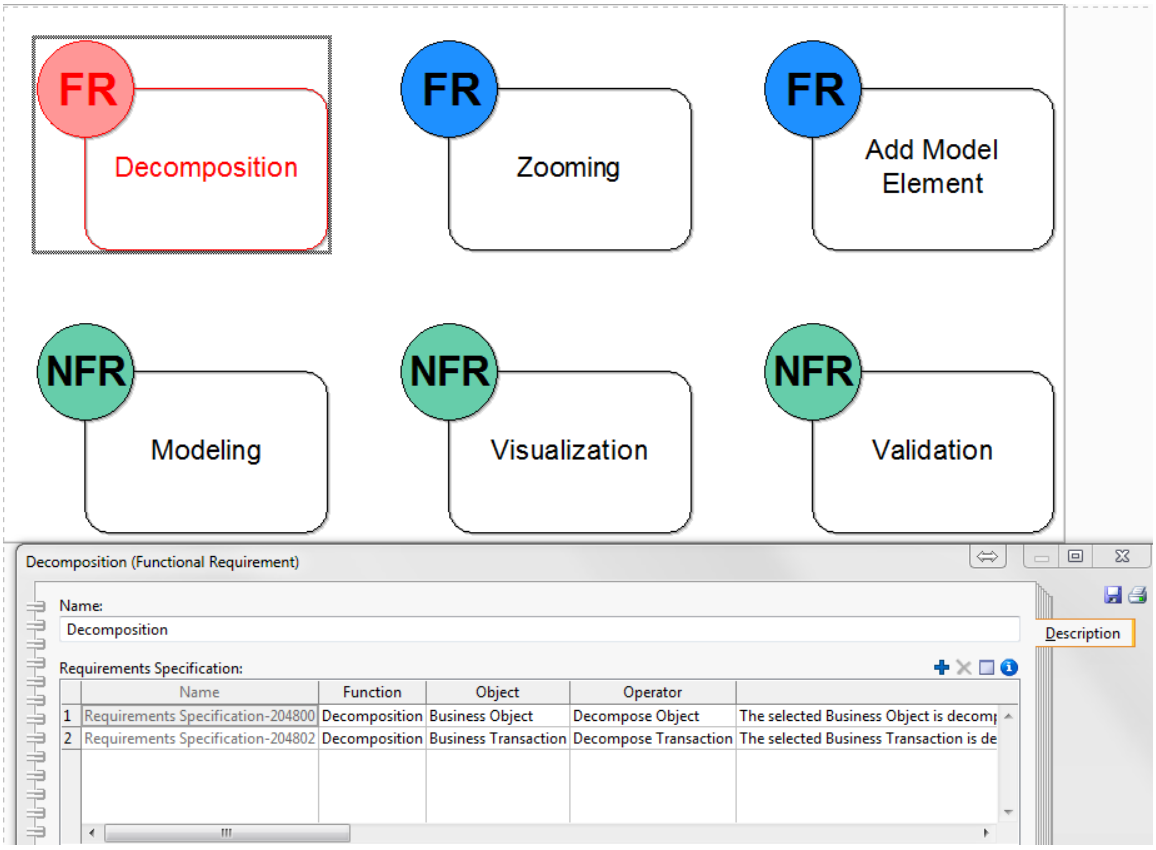


Figure 80: Excerpt of the SOM Conceptual Design model created with the MUVIEMOT tool

8.3.6 Modeling Procedure of the MUVIEMOT tool

The MUVIEMOT modeling method describes an idealistic procedure of applying the method (cf. section 6.3.7). The MUVIEMOT modeling tool, trying to serve different possible users, furthermore realized two application scenarios:

1. **The free scenario**
   In the free scenario, the MUVIEMOT user can basically do whatever he or she wants. There is no guidance in the way of transforming models of one step into initial models of a succeeding step. The user can decide when to create and integrate the multiple models of the MUVIEMOT method.

2. **The guided scenario**
   The second scenario fully supports the methodical direction of actions, i.e., the modeling procedure as introduced in section 6.3.7 by means of a concrete sequence of modeling steps. The tool's emphasis is to support the modeler in this scenario with the highest degree of automation possible. Hence, the tool provides transformation capabilities between modeling steps and semantically integrates the derived models. Therefore, consistency

between the multiple models is realized, enabling valid and efficient conceptual design specification with the tool.

In the following, the most important tool functionality, developed to support the guided application scenario, are briefly described.

Modeling Scenario to Modeling Language Transformation After finishing the *Modeling Scenario* step, the MUVIEMOT user can trigger a functionality that automatically generates empty *viewpoint model* models, names them according to the specification, stores them in the same ADOxx model group, and adds an intermodel reference, linking the viewpoint element of the modeling scenario to the complete viewpoint model. The same procedure can be executed to create initial *meta model* models based on the meta model elements, specified in the modeling scenario.

Viewpoint Model Specification When specifying the viewpoint models, the MUVIEMOT user is enabled to reuse the already specified modeling classes and relation classes of the *meta model* models. These concepts can be copied in the meta model model and pasted into the viewpoint model.

Modeling Language to Modeling Procedure Transformation When the MUVIEMOT user first opens a *Modeling Procedure* model, the tool asks whether it should search for viewpoint models and automatically generate the corresponding viewpoint elements in the triggered-in and effect-on areas of the modeling procedure model. If the user agrees, the tool creates all viewpoint elements, adds an intermodel reference to integrate the models, and positions them according to an auto-layout algorithms.

Modeling Language to Viewpoint Dependencies Transformation When a *Viewpoint Dependencies* model is opened for the first time by the MUVIEMOT user, the tool asks whether it should search for viewpoint models and automatically compute the syntactic dependencies. If the user agrees, the tool opens all viewpoint models and creates an inverted list with the modeling concept as a key and the list of viewpoints the concept is used in, as value. This list is then transformed into a graphical representation and, after automatically positioning all elements according to a layout algorithm, presented to the user for further processing. He or she may then add semantic dependencies or further constrain the syntactic dependencies. Each identified dependency is also transformed into a row of the *view dependencies* attribute value of the corresponding modeling concept, presented to the modeler by means of a table (cf. Figure 79 at the bottom).

Modeling Procedure to Conceptual Design Transformation When the MUVIEMOT user first opens a *Conceptual Design* model, the tool asks whether it should automatically

instantiate the model. If the user agrees, the tool opens all modeling procedure models and transforms all identified use cases into functional requirements and automatically positions them in an appropriate way on the modeling canvas of the Conceptual Design model.

8.4 Model-Driven Development of Multi-View Modeling Tools

The MUVIEMOT modeling environment not only supports the utilization of the MUVIEMOT method, a major benefit of creating the multiple models according to the method specification is the possibility to transform the codified knowledge into an initial running prototype of the specified multi-view modeling tool. The models therefore not only serve as a specification and communication means, they moreover enable model-driven development of modeling tools. This functionality enables a significant increase of efficiency concerning not only tool design but especially tool development.

The model-driven development approach is strictly implemented for the ADOxx meta modeling platform. Hence, the MUVIEMOT models are transformed into an ADOxx Library Language (ALL) specification format (referred to in the following as ALL specification). This ALL specification can then be transformed into an ADOxx Application Library (ABL) (referred to in the following as ABL library) using the *ALL2ABL webservice*[47] of the BOC AG. The generated library can be imported into the ADOxx Development Toolkit. Afterwards, the ADOxx Modeling Toolkit can be used to immediately create multi-view models according to the MUVIEMOT conceptual design specification. The major steps in the transformation algorithm are as follows:

- Viewpoint Models are transformed into ADOxx modeltypes (see Figure 81).
- The attributes, graphical visualizations and attribute representations, specified for modeling classes and relation classes in viewpoint models are considered and realized as attributes of the resulting concepts in the ADOxx meta model.
- Inheritance relationships between modeling classes are reflected by accordingly specified inheritance relationships between the concepts in the generated ADOxx meta model.
- Syntactic dependencies between attributes or complete concepts, specified in the Viewpoint Dependencies model are transformed into consistency preserving algorithms using AdoScript code. These algorithms ensure consistent modeling with multiple views. Hence, e.g., changes of attribute values are recognized by the script. The script then checks, depending on the information about the viewpoint dependencies, which modifications in other viewpoints need to be executed in order to preserve consistency. Finally,

[47] The ALL2ABL webservice can be found at: `http://www.adoxx.org/live/adoxx-development-tools` last checked: 2015-02-24

the scripts executes all necessary modifications in the background, therefore realizing multi-view modeling by design.

- Functional requirements can be appended with a pseudo code or AdoScript code implementation of the respective functionality. These code snippets are imported and triggered by the corresponding events of the platform. Hence, the tool developer can use the pseudo code as a requirements specification.

The emphasis of the transformation is on the efficiency improvements gained through the model-driven development approach. The transformation results in a first prototype of the multi-view modeling tool the tool developer needs to refine using the built-in functionality of the ADOxx meta modeling platform.

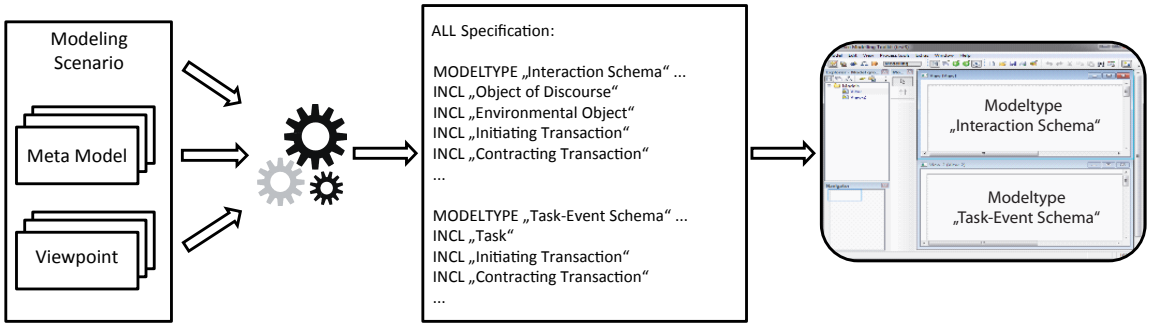


Figure 81: Transforming the Modeling Scenario model into ADOxx modeltypes (BORK AND KARAGIANNIS, 2014)

8.5 Evaluation

Evaluation of the MUVIEMOT modeling environment is performed in two ways: First, section 8.5.1 elaborates on an illustrative scenario of applying the tool by means of modeling a conceptual design for a SOM business process modeling tool and generating an initial prototype of the tool following the model-driven development approach introduced earlier. Second, in section 8.5.2, the functionality of the realized MUVIEMOT tool is contrasted to the identified requirements.

8.5.1 Case Study: Model-Driven Development of a SOM Tool

In order to evaluate the utility of the MUVIEMOT modeling environment and especially the model-driven development approach, the conceptual design specification of a SOM business process modeling tool already used throughout this section has been transformed into an initial prototype implementation. The created prototype has been tested for correctness according the the method specification and the intended outcomes.

In the case study, lots of effort has been put in the conceptual design specification in order to elaborate on the whole transformation capabilities of the model-driven development approach. For each modeling class and relation class of the multiple SOM business process viewpoints, the graphical visualization by means of a *GraphRep* and the attribute representation by means of an *AttrRep* has been specified using the attributes of the elements in the respective viewpoint models. Inheritance relationships like the one between the abstract class business object and its subclasses object of discourse and environmental object have been codified. Moreover, lots of attributes, not only for modeling classes and relation classes, but also for viewpoint models have been specified. The latter type of attributes are transformed into attributes of the ADOxx modeltype. An emphasis of the case study was to evaluate the consistency mechanisms for the syntactic dependencies identified by the tool.

As the SOM method utilizes multi-view modeling by design, the conceptual design also included coverage of synchronization mechanisms. Hence, it was specified, that after the modeler creates a new SOM business process model, all four viewpoints should be instantiated and visualized simultaneously to the modeler. Moreover, AdoScript code has been added that should realize the event handling after a new element has been created, an element has been deleted, or an attribute has been changed.

Lessons Learned

The model-driven development worked perfectly as supposed to. The generated viewpoints have been transformed into according ADOxx modeltypes. The relationships between modeling classes and relation classes as well as the inheritance relationships have been transformed into an ADOxx meta model. The event handling worked as supposed, i.e., after creating a new SOM business process model, the prototype automatically generated four views and showed them with an appropriate position. Moreover, the tool recognized attribute value changes and triggered the code to ensure consistency between the multiple SOM views. For example, creating a new environmental object in the Object Decomposition Schema forced the automatic generation of an environmental object also in the Interaction Schema. Deletion of elements has also been handled.

Due to the experience of implementing a SOM modeling 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 performed. MUVIEMOT and the generation of the ALL specification extremely foster the conception and implementation of multi-view modeling tools on a higher abstraction level. Method engineers can concentrate on the domain-concepts and define the conceptual design of a multi-view modeling tool in an intuitive way. They don't have to elaborate on technical details e.g., how to define a meta model in a specific tool development environment or how to generate multi-view modeling editors. Due to the automatic

transformation, lots of the functionality can be generated automatically. Tool developers can then concentrate on the very specific requirements of a modeling method that cannot be generated or implemented automatically.

In sum, comparing the effort required to model all discussed aspects in the MUVIEMOT tool with the implementation of the meta models and the synchronization mechanisms from scratch approved the assumed increase of development efficiency gain. However, depending on the complexity of the multi-view modeling method, additional implementation needs to be done. This implementation essentially regards the modeling procedure and the mechanisms & algorithms of the method. Consequently, the generation limits the effort for implementation to a minimal level. In case of the SOM tool, due to its complex modeling procedure, the transformation didn't result in a modeling tool that could be published for e.g., teaching the SOM method. However, on a syntactic point of view, not considering usability, the tool was conforming to the SOM meta model and allowed rudimentary multi-view creation and editing of SOM business process models using drag & drop functionality. Notably, the SOM MUVIEMOT models have been transformed into over 3000 lines of ALL specification.

8.5.2 Requirements Analysis

Besides the technical realization evaluation described in the preceding section, a further evaluation of the MUVIEMOT tool is performed by contrasting the functionality of the current version of the MUVIEMOT tool with the requirements defined in section 8.2. Table 31 summarizes the requirements and their respective fulfillment.

Table 31: MUVIEMOT tool requirements evaluation

ID	Value	Description
MUVIEMOT Requirements		
MVMT-1	●	The tool provides a modeltype for each step of the MUVIEMOT method, specifically designed to capture the aspects relevant for the step.
MVMT-2	●	The tool realized individual graphical notations for each concept of the MUVIEMOT method.

MVMT-3	●	The tool provides for each concept of the MUVIEMOT method a reasonable set of attributes the modeler can view and edit using an ADOxx Notebook.
MVMT-4	●	An emphasis of the MUVIEMOT tool was on the appropriate support of the method's modeling procedure. The tool provides not only a guided application scenario with automatic transformations between MUVIEMOT steps, it also provides a free application scenario, hereby letting the modeler decide when to create which model.
MVMT-5	◐	Consistency is guaranteed if the modeler agrees to follow the guided application scenario. Consequently, if the free application scenario is chosen, consistency between the different MUVIEMOT steps depends on the capabilities of the modeler.
		Application Domain Requirements
APPD-1	●	Viewpoint specification with the MUVIEMOT tool is twofold: First, an informal specification is performed in the Modeling Scenario step by relating the viewpoint to the meta models by means of *projection* or *selection* relationships. Second, a Viewpoint Model is provided to the modeler. With this model, modeling classes, relation classes, and attributes can be specified.
APPD-2	●	The tool provides a specialized modeltype for the creation of Multi-View Modeling Use Case models. The tool also provides an automatic positioning algorithm for ensuring intuitive understanding and efficient creation of the models.
APPD-3	●	The tool provides an automatic generation and graphical visualization of syntactic viewpoint dependencies. Semantic viewpoint dependencies can be added by the modeler in an intuitive way.

APPD-4	○	The tool, in its current version, provides no stable transformation generation mechanisms. Notably, the challenge is not a graphical notation for the specification of these transformations. By contrast, the challenge is how to transform theses graphical models into AdoScript code that can be applied to any instance of a source modeltype into an instance of a target modeltype. This feature is left for future work.
APPD-5	●	The tool provides a visualization for the viewpoint dependencies. Moreover, the syntactic dependencies can be generated automatically.
APPD-6	●	An emphasis of the tool has been on the model-driven development approach. The tool enables the automatic transformation of a conceptual design into initial prototype implementations of a corresponding multi-view modeling tool based on ADOxx.
APPD-7	◐	The syntactic viewpoint dependencies between concepts and attributes are transformed into AdoScript code. The code is written on meta model level, therefore, enabling its applicability for any model created in the generated modeling tool. However, complex semantic dependencies are not considered in the transformation process, yet. This implementation task is left for future work.
APPD-8	◐	The tool provides a viewpoint-specific definition of graphical visualizations. However, in the current version of the ADOxx platform, their realization in the platform can be considered a workaround only.
		Non-Functional Requirements
NFR-1	●	The tool provides two application scenarios, one that is aligned to the MUVIEMOT modeling procedure, and one that provides the modeler with the possibility to create the models in any order he or she prefers.
NFR-2	●	The tool is published on a webpage, along with several documents, comprising an introduction, tutorial videos, handbooks, and download & installation instructions.

NFR-3	●	The tool provides tool tips and/or help texts for every interesting modeling concept and attribute. This additional information is comprised and presented to the modeler in ADOxx Notebooks.
NFR-4	◐	If the modeler follows the guided application scenario, the tool automatically recognizes changes and performs according modifications in multiple MUVIEMOT models in order to keep them consistent. However, in the free application scenario, it depends on the modeler whether these synchronization mechanisms can be executed as they require certain attributes to be filled with intermodel references. These intermodel references, set automatically in the guided application scenario, need to be set by the modeler manually in the free application scenario.

The current version of the MUVIEMOT modeling environment has being implemented as a project within the Open Models Initiative (KARAGIANNIS ET AL., 2008) using the facilities of the Open Models Laboratory (OMiLAB)[48] at the University of Vienna. The prototype is accessible as open source and open use platform[49].

8.6 Summary

The previous section described the conception and implementation of a modeling tool, supporting the MUVIEMOT method. This section started with a collection of requirements for such an modeling tool. Thereafter, the MUVIEMOT modeling environment has been introduced by relating each MUVIEMOT step to the correspondingly implemented ADOxx modeltype. Afterwards, the interplay of these modeltypes by means of a modeling procedure has been described. Two application scenarios have been identified and implemented for the tool: a *free application scenario* and a *guided application scenario*.

Evaluation of the tool has then been performed twofold: First, a case study concentrated on the model-driven development capabilities of the tool by transforming the SOM conceptual design into a SOM multi-view modeling tool. Second, the defined requirements have been compared to the functionality and constituents of the implemented MUVIEMOT tool.

The generated models contribute to the coordination between method engineers and tool developers, aiming to bridge the inherently given abstraction gap. Opportunities to further improve the capabilities of the tool have been identified but are left for future work. Also,

[48] OMiLAB homepage: `http://www.omilab.org`, last checked: 2015-02-22
[49] MUVIEMOT project page: `http://www.omilab.org/web/muviemot/home`, last checked: 2015-02-24

improvements of the MUVIEMOT method require corresponding changes or addition of new functionality to the modeling tool. The tool needs to be used in a large number of cases in order to derive a mature and stable version. The plan is to use the tool in future master's degree meta modeling courses at the University of Vienna. This would enable a qualitative evaluation of the tool and might result in a number of interesting ideas for further improvements of both, the MUVIEMOT method and the MUVIEMOT tool. Figure 82 illustrates the overarching architecture of the implemented tool. Releases notes of different versions of the MUVIEMOT tool can be found in Appendix B.

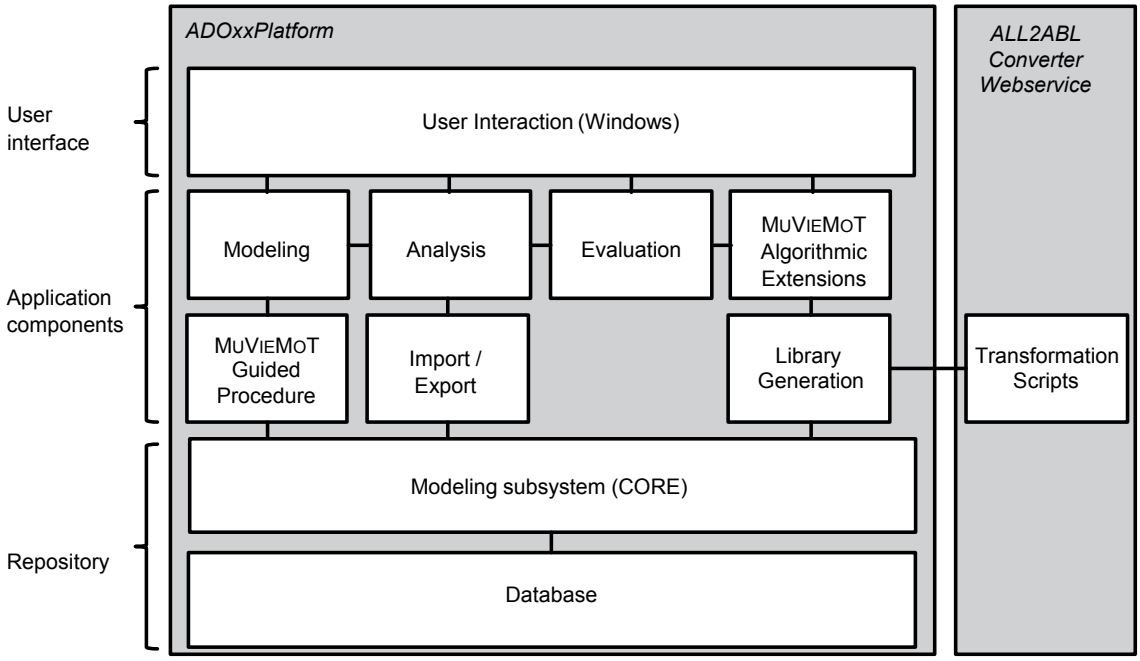


Figure 82: MuVieMoT tool architecture (BORK AND KARAGIANNIS, 2014)

9 Conclusion & Future Work

The aim of this section is to summarize the main contributions of the thesis (section 9.1) and to point the reader to directions of future work (section 9.2).

9.1 Conclusive Remarks

Modelers of complex systems like enterprise systems, information systems, or software systems face the problem of the increasing complexity of the reality that needs to be mapped into a formalized representations, i.e. models. The thesis at hand investigated the possibilities of utilizing multi-view modeling to cope with this inherent complexity by decomposing this overarching representation into multiple views. Hence, modelers are provided with decomposed subareas to be captured in a view. The combination of the views then gives the whole model of the system under study. As the views have syntactically and/or semantically overlapping areas, consistency between them not only needs to be considered in a static manner but also in a dynamic manner. Consequently, the aim of this thesis was to explore the fundamentals of multi-view modeling from an meta modeling standpoint, focusing on the needs of method engineers and tool designers.

Following the theoretical foundation of multi-view modeling, the major contribution of this thesis is the proposition of the conceptual modeling method MuVieMoT. The method comprises several steps, guiding method engineers in specifying and tool developers in realizing a conceptual design for a multi-view modeling tool. The aim of the MuVieMoT method is to bridge the semantic gap between method owners or method engineers on the one side and tool developers on the other in a procedural way. In this regard, each step of the MuVieMoT modeling procedure provides a dedicated conceptual modeling languages that has been carefully designed to address the specific needs of the corresponding step. The goal of the designed modeling languages is to provide better means in capturing the specifics of multi-view modeling methods on a more suitable abstraction level. However, there is no measure to quantify the degree of fulfillment of this goal (cf. (Houy et al., 2012; Stark and Esswein, 2012)). The method is generically designed, hence, it can be applied for integrating arbitrary modeling languages in a multi-viewing manner. With each step of the approach, the focus is shifting from a very abstract and overarching specification of a multi-view modeling scenario towards concrete requirements for the development of a multi-view modeling tool.

The MuVieMoT method should be utilized to ease the requirements specification for multi-view modeling tools with an emphasis on consistency between the views and the way modelers interact with them. The method has been implemented on the ADOxx meta modeling platform, enabling free access to a modeling environment, allowing tool-supported application of MuVieMoT. The tool not only realizes model types for each step of the method, it also provides

model transformations in order to increase efficiency. Moreover, it provides mechanisms to transform the complete conceptual design specification into a multi-view modeling tool, realized on the ADOxx meta modeling platform. Hence, the knowledge codified in the conceptual models can be used for model-driven development of initial, consistency-preserving multi-view modeling tools.

"Additions to the knowledge base as results of design science research will include any extensions to the original theories and methods made during the research, the new meta-artifacts (design products and processes), and all experiences gained from performing the research and field testing the artifact in the application environment" (HEVNER, 2007, p. 90). Following the design science research perspective, the MUVIEMOT method can be classified as of category *method*, an artifact concerned with prescriptive knowledge (cf. (GREGOR AND HEVNER, 2013) and (VAN AKEN, 2005)). Moreover, following the categorization along the dimensions *application domain maturity* and *solution maturity* presented by GREGOR AND HEVNER (2013), the MUVIEMOT method can be considered an *improvement*. Improvements in design science research are characterized as an achievement by developing *"new solutions for known problems"*(GREGOR AND HEVNER, 2013, p. 345).

Notably, the MUVIEMOT modeling environment is not the only tool that has been developed during the work on this thesis. As a side-product, during the case studies of the thesis, a multi-view modeling tool for the Semantic Object Model (SOM) enterprise modeling method has been developed on the ADOxx meta modeling platform. The SOM tool is released as a free download. The tool is used at three universities in enterprise modeling and business process modeling lectures.

Evaluation was a major issue in this thesis. For all three artifacts presented, i.e., the MUVIEMOT method, the MUVIEMOT modeling environment, and the SOM enterprise modeling tool, evaluation was performed using illustrative scenarios as proposed by PEFFERS ET AL. (2012). Hence, all artifacts have been tested in real-world situations, hereby proving their usability and utility. *"The development of elaborate and carefully documented artifacts (e.g., industry reference models and modeling methods) requires resources that even well-equipped research groups at universities do not have at their disposal"* (FRANK ET AL., 2014, p. 2). The evaluation therefore emphasized on the practicability and utility of the artifacts, omitting questionnaires and qualitative evaluation approaches due to the unsatisfying relation between effort necessary for their realization and the gained insights (cf. (FRANK, 2011b, p. 97)). Consequently, there is of course some personal bias left in the proposed method and its constituents; and some ambiguity in the graphical visualizations (FRANK, 2002, p. 82). To solve this, the method needs to evolve during multiple instantiations, leading to ideas for further refinements and improvements.

9.2 Future Work

The thesis at hand does not aim to solve the multi-view modeling problem completely. However, it provides a first set of ideas, concepts, methods, and tools that help method engineers during the conceptualization process. During the work on this thesis, several ideas came up, that, due to limited time, could not have been finalized yet. Hence, future work directions are briefly discussed in the following.

One possible direction for future research is to provide better means of visualizing viewpoint dependencies in the multi-view modeling method setting. Graphviz (ELLSON ET AL., 2002, 2004), a tool that has, besides numerous other domains, also proven to be beneficial in visualizing the dependencies of software artifacts, could be worth investigating. A different approach has been published by QAMAR ET AL. (2013) who proposes not only a domain-specific language for the specification of dependencies between ecore-based multi-view systems engineering, but also a graphical visualization, automatically generated out of the ecore models. Notably, Qamar realizes the visualization by utilizing the DOT language of Graphviz (GANSNER ET AL., 2010). Looking at the concepts and adopting the functionality in the MUVIEMOT tool seems promising towards increasing utility of the tool.

In the current MUVIEMOT version, consistency between the views is guaranteed by the modeling environment. However, in some scenarios, it may be beneficial to allow temporary inconsistencies e.g., in order to increase usability and creativity of working with a tool. It might therefore be worth investigating to translate the view dependencies of MUVIEMOT into validation checks. The generated modeling tool would then allow inconsistencies. The modeler is able to model without any constraints. After he or she has finished the modeling process, a model validation algorithm can be executed. This algorithm, initialized with the viewpoint dependencies, searches for inconsistencies in the multiple views and highlights the findings appropriately.

Another interesting direction of future research is the transformation of the theoretical foundation of multi-view modeling methods and multi-view modeling tools into generic requirements for a meta modeling platform component, enabling to easily configure and plug-in multi-viewing functionality for arbitrary modeling methods. It is desirable to create configuration functionality, enabling to efficiently customize the *by design* and *by generation* multi-view modeling approaches for any modeling method at hand. Hence, the research is targeting at conceptualizing a generic multi-viewing component for meta modeling platforms.

Appendix A Implementation of a SOM Multi-View Modeling Tool

This section describes the development of the SOM multi-view modeling tool in more detail. The SOM tool is not the center of the thesis, however, it is an outcome of working on the thesis. Therefore, it seems appropriate to elaborate also on the implemented tool.

Appendix A.1 Motivation & Background

Starting in the 1990s, several modeling tools supporting the Semantic Object Model have been developed, e.g., *SOM V3* and *SOMpro*. All tools utilized different approaches in the way SOM multi-view models should be modeled; have been realized with different programming languages and development environments; covered different aspects of the SOM method (depending on the maturity of the SOM method itself at the time the tool development took place); have been part of the work of different phd students. Although suitable from a functionality perspective, all tools lacked at being adoptable to the continuous improvements of the SOM method. As the phd students left the university, the intellectual entry barrier for a new developer has been too huge. Hence, one of the goals of the SOM modeling tool, developed by the author of this thesis, was to use an established meta modeling platform that is commonly used by developers around the world, therefore promising durability and maintenance support.

This motivation led to the decision of using the ADOxx meta modeling platform. ADOxx has been successfully used for the development of manifold modeling tools (see an excerpt of them on the Open Models Laboratory project overview webpage[50]). The first prototype of the tool was part of a working package of the forFLEX[51] research project. The work on the tool therefore started with a two day platform training held at the facilities of the University of Vienna. Following this compact training, conception and development of the SOM tool was performed autonomously by the author of this thesis while working as a university assistant at the University of Bamberg.

The development was led by Prof. Dr. Elmar J. Sinz. The coordination of the development tasks, the conception of the multi-view modeling tool, major implementation tasks, and project management have been performed by the author of this thesis. Four bachelor students have significantly contributed in the tool development stage. Table 32 summarizes the involved people, and their main contributions. As almost every work package was resolved by groups of two students, the student teams are accordingly grouped in the table.

[50] OMiLAB project overview webpage: `http://www.omilab.org/web/guest/projects`, last checked: 2014-12-12

[51] forFLEX research project webpage: `http://www.forflex.de/index.php`, last checked: 2014-12-12

Table 32: SOM modeling tool development team

CONTRIBUTOR	CONTRIBUTION
Elmar J. Sinz	Professor at the University of Bamberg. Leader of the development team. Co-leader of the forFLEX research project and co-inventor of the SOM modeling method.
Dominik Bork	University assistant at the University of Bamberg. Project coordinator and chief developer. Responsible for all aspects considering conceptualization, coding, testing, integration of code snippets implemented by the students, and project management.
Andreas Steffan & Michael Stretz	Bachelor students at the University of Bamberg. Andreas and Michael worked on the layout algorithms for SOM business process models, especially the *auto-layout* and the *smooth edges* layout algorithms of the Interaction Schema. Moreover the two started with the implementation of the transformation of SOM business process models into Schema of Task Classes and introduced rudimentary model validation functionality.
Felix Härer & Steffen Witt	Bachelor students at the University of Bamberg. Felix and Steffen worked on the transformation of SOM business process models to SOM business application system models (i.e., Schema of Conceptual Classes). Moreover, they worked on the decomposition of business objects and business transactions, the support for defining the behavior of SOM bp models in the Task-Event Schema, and initialized the development on the BPMN transformation.
Wilfrid Utz	Doctoral student at the University of Vienna and researcher at BOC Asset Management. Wilfrid considerably helped to deploy the SOM modeling tool on version 1.5 of the ADOxx platform. Moreover, he acted as a technical consultant, providing platform insights that have been inevitable for the successful conception and realization of tools, developed on ADOxx.

Appendix A.2 Releases of the SOM Business Process Modeling Tool

In the following, the releases of the SOM modeling tool are briefly described. For each release, the main improvements and important functionality that has been added, compared to the former release, are described.

Appendix A.2.1 Release I: 2010-12-15

The first release of the SOM tool was made publicly available in December 2010. The release covered the business process modeling part of SOM, i.e., it was constituted of four ADOxx model types: Interaction Schema, Task-Event Schema, Object Decomposition Schema, and Transaction Decomposition Schema. The simultaneous visualization of the multiple views and initial consistency management between the views have been included, e.g., changing the name of concepts in one view has resulted in accordingly performed changes to the other, affected views.

At that time, the tool was used for scientific purposes only. First publications, documenting on the conception of the SOM multi-view modeling tool (BORK AND SINZ, 2010, 2011a,b) have been realized and presented at international workshops, project meetings, and conferences. Moreover, the tool has been given to selected SOM experts in order to gain some feedback and ideas for further improvements.

Appendix A.2.2 Release II: 2012-09-12

In September 2012, the second prototype of the SOM tool has been published at the Open Models Initiative website. Compared to the first prototype, the functionality of the tool has been increase significantly. Now, also the Schema of Task Classes (TAS) and the Schema of Conceptual Classes (COS) have been realized as model types in ADOxx. The tool supported model-driven transformation of initial TAS and COS schemata from SOM business process models. Further processing of the generated models has been implemented using context menus. Additionally, research of PÜTZ AND SINZ (2010b), resulted in a model-driven transformation between SOM business process models and Business Process Model and Notation (BPMN) schemata. The second prototype of the SOM tool already included the functionality by implementing the meta model mapping.

Major improvements by means of functionality and utility of the tool have been realized. The second prototype introduced the restriction of functionality to the viewpoints, it could be executed at (according to the method specification). Hence, e.g., the functionality of executing an auto-layout algorithm was restricted to the Interaction Schema, and was not even visible to the modeler in the context menu of the other views. Moreover, zooming functionality has

been implemented and contrasted to the decomposition functionality. Using the different decomposition levels of business objects and business transactions, the modeler was enabled to switch immediately between already defined SOM business process levels. In order to define a higher decomposition level or to remove the currently defined bp level, the modeler was forced to execute the *increase level of business process* or *decrease level of business process* functionality, respectively. Consequently, zooming in and out of the business process did not change the stored connections between business objects and business transactions. Zooming only restored and visualized a formerly defined bp level.

Research in the context of the forFLEX research project, primarily performed by WAGNER AND FERSTL (2010), resulted in the requirement of introducing contextual information to SOM business process models. The ADOxx meta model has been extended accordingly, as well as the synchronization scripts between the multiple SOM bp models. The tool also provided initial validation algorithms, enabling to test SOM models for inconsistencies, e.g., considering attribute values of synchronized objects, or the comprehensive specification of the SOM bp behavior in the Task-Event Schema.

The second SOM prototype was used at the University of Bamberg in lectures held by Prof. Sinz and supported by the author of this thesis, e.g., in bachelor seminars. The feedback gained from using the tool with students, who have not been experts in the SOM method, resulted in further requirements and improvement ideas for the third release.

Appendix A.2.3 Release III: 2014-11-18

In 2014, release three of the SOM modeling tool has been published. The third release had many improvements tackling usability complaints made by the students who used the second release in university courses. Moreover, the SOM tool was the first modeling tool realized on version 1.5 of ADOxx.

The evaluation of the former releases revealed two major weaknesses: First, the SOM tool, if utilized as intended by the method, has proven to be very useful for the creation of multi-view business process models and business application systems models. Second, the majority of all negative feedback was concerned with installation problems due to the SQL database server. Thus, the emphasis of the development was not to integrate more functionality. By contrast, the goal was to prevent modelers from utilizing the tool in a way that is not intended by SOM and therefore leads to inconsistencies and invalid model states. The most notable improvements in this direction have been:

Prevent deletion of Business Objects and Business Transactions Extensive extensions to the ADOxx event handling have been implemented to prevent manual deletion of business objects and business transactions by the modeler. Therefore, a script checks,

whether the deletion is performed by the platform or by the modeler. In the latter case, all business objects and business transactions are restored, all attribute values are restored, and the modeler is prompted a warning to make him/her aware of the misleading usage of the tool.

Prevent drag & drop modeling Usage of former SOM releases by students revealed the fact, that most modelers intuitively try to adopt drag & drop modeling, i.e., selecting modeling concepts (objects or relations) in the palette and placing them somewhere on the modeling canvas. However, SOM modeling requires the recursive application of the specified decomposition rules in order to refine an initial SOM bp model. Therefore, the palette of modeling concepts has been reduced to a minimum. Modelers are prevented from selecting modeling objects and placing them on the modeling canvas in the first place.

Improve the installation routine Release III of the SOM tool has been built on version 1.5 of the ADOxx platform. The change of the development platform also enabled the utilization of the new ADOxx installation routine. With this routine, only one installer needs to be executed. The required SQL server version is automatically detected, downloaded, and installed. If there is already a suitable installation of the SQL server, only a new database instance for the SOM tool is generated. Moreover, migrating the SOM tool to version 1.5 of the platform enabled installation and usage of the tool with Windows 8 and 8.1 operating systems.

Appendix A.2.4 Summary

Downloads of the SOM modeling tool has tracked starting autumn 2014. However, only downloads from the official Open Models Laboratory project page of SOM have been tracked. In total, release III of the SOM tool has been downloaded almost 50 times since November 2014. Moreover, the tool has been used in lectures at the University of Bamberg, the University of Vienna, and the Virtual University of Bavaria[52].

[52] Virtual University of Bavaria (VHB), `http://www.vhb.org/en/homepage/`, last checked: 2014-12-14

Table 33: Releases of the SOM modeling tool

NO	DATE	MAJOR FEATURES
I	2010-12-15	Multi-view modeling of SOM business process models.
II	2012-09-12	Model-driven derivation of COS, TAS, and BMPN models. Implementation of increase/decrease of business process level and zooming in/out of business process level functionality. Enable the modeler to attache contextual information to business objects, business transactions, and tasks. Integration of pre- and post-conditions to the tasks within the Task-Event Schema. Initial efforts to validate SOM bp models. Import/Export of SOM models. Usability improvements. Added customization support, e.g., changing the default color of business objects, business transactions, and tasks.
III	2014-11-18	Consolidation of COS and TAS models. Major usability improvements, i.e., prohibit the modeler from utilizing the SOM tool in a way that is not conforming with the method. Prevent manual deletion of modeling classes and relation classes. Prevent drag & drop modeling. Upgrade to ADOxx version 1.5.

Appendix B Implementation of the MuVieMoT Modeling Tool

This section summarizes the road map towards the development of the MuVieMoT modeling environment. The tool has been developed as project within the Open Models Initiative Laboratory (OMiLAB) and can be downloaded with lots of additional information, e.g., project details, references, video tutorials, case studies, from the corresponding project webpage[53].

Appendix B.1 Releases of the MuVieMoT modeling tool

In the following, the implementation of the MuVieMoT modeling tool is described briefly. Table 34 summarizes the different prototypes of the MuVieMoT modeling tool together with some release notes. Section B.1.1 until section B.1.4 then briefly describe the major changes introduced with each new prototype version of the MuVieMoT environment.

Table 34: Releases of the MuVieMoT modeling environment

No	Date	Features
1	2014-08-20	Model-based specification of conceptual designs for multi-view modeling tools. Realized on ADOxx 1.3.
2	2014-09-20	Model-driven development of multi-view modeling tools with MuVieMoT. Integration of the ALL2ABL web service, hosted by the BOC AG. Knowledge, modeled in the viewpoint dependencies model are transformed into AdoScript code that ensures consistency between the multiple views.
3	2015-01-07	Mature version of MuVieMoT released. Added more complex view dependencies and the corresponding AdoScript code for the model-driven development procedure.

[53] MuVieMoT project webpage, http://www.omilab.org/web/muviemot/home, last checked: 2015-04-15

4	Autumn 2015	New MUVIEMOT version, based on ADOxx 1.5 released. Major improvements have been made according to the transformation of the viewpoint models into synchronization algorithms in AdoScript. Minor improvements have been realized targeting at usefulness and intuitiveness of the tool. With this release a lot of documentation material has been published like case studies and tutorial videos.

Appendix B.1.1 MUVIEMOT Release I: 2014-08-20

Version I of the MUVIEMOT modeling tool can be considered a minimal implementation of the MUVIEMOT method. It comprised ADOxx model types for each of the modeling procedure's steps. However, the multiple model types, and the models created accordingly, have not been integrated. Changes performed in one model have not been reflected in other models. Moreover, consistency was not checked by the tool automatically. Version I of the tool was based on ADOxx version 1.3.

Appendix B.1.2 MUVIEMOT Release II: 2014-09-20

The development of the version II of the MUVIEMOT was focused on the modeling procedure of the MUVIEMOT method. Therefore, model transformations between adjacent modeling steps of the approach have been implemented in the tool wherever suggested by the method. Targeting at efficiency gains, additional functionality not specified in the method but inspired by feedback of users of version I of the tool, have been realized. The ADOxx Notebooks of the viewpoint and meta model modeling classes have been extended. Consequently, modelers have been enabled to create empty viewpoint models and meta model models by executing the functionality within the Notebook of the concepts in the modeling scenario model.

Another emphasis of the second release was the integration of the *ALL2ABL converter service* hosted by the BOC[54] into a comprehensive multi-view modeling tool development environment. Rudimentary code generation functionality has been realized, enabling the model-driven generation of ADOxx modeltypes according to the MUVIEMOT viewpoint models. For each viewpoint model one ADOxx modeltype has been created, including all modeling classes and relation classes as well as the inheritance relationships between modeling classes.

The ADOxx Notebooks of modeling classes and relation classes have been extended in order to incorporate attributes for specifying the Notebook and the graphical representation of a

[54] The ALL2ABL converter service is a web service implemented and hosted by the BOC AG, `http://www.adoxx.org/live/adoxx-development-tools` last checked: 2015-03-20

modeling concept in the specific viewpoint (i.e., the resulting ADOxx modeltype).

Appendix B.1.3 MUVIEMOT Release III: 2015-01-07

The focus of the third release was twofold: First, major revisions on the existing implementation have been performed. Second, new functionality has been added that enabled the automatic generation of synchronization scripts in AdoScript. These scripts have been incorporated in the model-driven development of multi-view modeling tools. Hence, the dependencies modeled in the viewpoint dependencies model of the MUVIEMOT tool have been processed in order to realize mechanisms on meta model level that ensure consistent multi-view modeling with the generated modeling tool.

The scripting mechanisms ensured, that concepts, i.e., modeling classes and relation classes, that follow a 1:1 mapping are automatically generated and attribute changes are synchronized. Hence, event handlers have been realized for recognizing the modeling operations performed by the modeler. After the modeler adds a new concept to the modeling canvas of a view that is included in the synchronization, a script is executed that uses the modeled viewpoint dependencies to elaborate the necessary changes in other views and triggers them automatically.

Appendix B.1.4 MUVIEMOT Release IV: Autumn 2015

The forth release of the MUVIEMOT tool realized the already defined functionality of the third release on version 1.5 of the ADOxx platform. This enabled the usage of the more convenient installer that runs also on Windows 8.1 systems. Together with the new version, screencast videos have been uploaded to the MUVIEMOT project page in order to describe the intended usage of the tool to potential users[55].

[55] MUVIEMOT screencast videos: `http://www.omilab.org/web/muviemot/home`, last checked: 2015-04-09

Appendix C Publications

Table 35 summarizes the published articles of the author. The first three publications, realized from 2010 until 2013, have been written under the supervision and with the help of Prof. Dr. Elmar J. Sinz while working as a university assistant at the University of Bamberg.

Since 2013, the author of this thesis was employed as an university assistant at the University of Vienna. All publications since 2014 have been written under the influence and with the help of the Research Group Knowledge Engineering, led by Prof. Dr. Dimitris Karagiannis.

Table 35: Summary of publications

AUTHOR	TITLE	YEAR	OUTLET
Domenik Bork, Elmar J. Sinz	Design of a SOM Business Process Modelling Tool based on the ADOxx Metamodeling Platform	2010	4^{th} International Workshop on Graph-Based Tools (GraBaTs'2010)
Domenik Bork, Elmar J. Sinz	Ein Multi-View-Modellierungswerkzeug für SOM-Geschäftsprozessmodelle auf Basis der Meta-Modellierungsplattform ADOxx	2011	Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse
Domenik Bork, Elmar J. Sinz	Bridging the Gap from a Multi-View Modelling Method to the Design of a Multi-View Modeling Tool	2013	Enterprise Modelling and Information Systems Architectures (EMISA) - An International Journal, 8 (2)
Domenik Bork, Hans-Georg Fill	Formal Aspects of Enterprise Modeling Methods: A Comparison Framework	2014	47^{th} Hawaii International Conference on System Sciences (HICSS'2014)
Domenik Bork	Model-Driven Development of Multi-View Modeling Tools	2014	Phd poster session at Austrian Computer Science Day 2014 (ACSD'2014)

Domenik Bork, Dimitris Karagiannis	Geschäftsprozessmodellierung mit BPMN	2014	wisu - Das Wirtschaftsstudium 6/2014
Domenik Bork, Dimitris Karagiannis	Model-Driven Development of Multi-View Modeling Tools: The MUVIEMOT Approach	2014	9^{th} International Conference Software Paradigm Trends (ICSOFT-PT'2014)

References

[AALST ET AL. 2002] AALST, Wil Van D.; DESEL, Jörg ; KINDLER, Ekkart: On the semantics of EPCs: A vicious circle. In: NÜTTGENS, Markus (Eds.); RUMP, Frank J. (Eds.): *Proceedings of the EPK 2002: Business Process Management Using EPCs*, Gesellschaft für Informatik, Bonn, 2002, pp. 71–80

[ABRIAL 1980] ABRIAL, Jean-Raymond: *Specification Language Z: Basic Library*. Oxford University Computing Laboratory, 1980

[AKEHURST AND KENT 2002] AKEHURST, David; KENT, Stuart: A Relational Approach to Defining Transformations in a Metamodel. In: JÉZÉQUEL, Jean-Marc (Eds.); HUSSMANN, Heinrich (Eds.) ; COOK, Stephen (Eds.): *UML 2002 - The Unified Modeling Language* Bd. 2460. Springer Berlin Heidelberg, 2002. – ISBN 978–3–540–44254–7, pp. 243–258

[AKEHURST ET AL. 2003] AKEHURST, David; KENT, Stuart ; PATRASCOIU, Octavian: A relational approach to defining and implementing transformations between metamodels. In: *Software and Systems Modeling* 2 (2003), No. 4, pp. 215–239. – ISSN 1619–1366

[AKEHURST 2004] AKEHURST, David H.: Proposal for a Model Driven Approach to Creating a Tool to Support the RM-ODP. In: *Proceedings of the 1st International Workshop on ODP in the Enterprise Computing (WODPEC 2004)*, 2004

[ALDAWUD ET AL. 2003] ALDAWUD, Omar; ELRAD, Tzilla ; BADER, Atef: UML Profile for Aspect-Oriented Software Development. In: *The Third International Workshop on Aspect Oriented Modeling*, 2003

[ALTER 2008] ALTER, Steven: Defining information systems as work systems: implications for the IS field. In: *European Journal of Information Systems* 17 (2008), No. 5, pp. 448–469

[AMELUNXEN ET AL. 2008] AMELUNXEN, Carsten; KLAR, Felix; KÖNIGS, Alexander; RÖTSCHKE, Tobias ; SCHÜRR, Andy: Metamodel-based Tool Integration with Moflon. In: *Proceedings of the 30th International Conference on Software Engineering*. New York, NY, USA : ACM, 2008 (ICSE '08). – ISBN 978–1–60558–079–1, pp. 807–810

[ATKINSON AND KÜHNE 2003] ATKINSON, C.; KÜHNE, T.: Model-Driven Development: A Metamodeling Foundation. In: *IEEE Software* 20 (2003), Sept, No. 5, pp. 36–41. – ISSN 0740–7459

[ATKINSON ET AL. 2013a] ATKINSON, Colin; GERBIG, Ralph ; TUNJIC, Christian: 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. New York, NY, USA : ACM, 2013 (VAO '13). – ISBN 978–1–4503–2070–2, pp. 2:1–2:9

[ATKINSON AND STOLL 2008] ATKINSON, Colin; STOLL, Dietmar: Orthographic Modeling Environment. In: FIADEIRO, José L. (Eds.); INVERARDI, Paola (Eds.): *Fundamental Approaches to Software Engineering* Bd. 4961. Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–78742–6, pp. 93–96

[ATKINSON ET AL. 2010] ATKINSON, Colin; STOLL, Dietmar ; BOSTAN, Philipp: Orthographic Software Modeling: A Practical Approach to View-Based Development. In: MACIASZEK, Leszek A. (Eds.); GONZÁLEZ-PÉREZ, César (Eds.) ; JABLONSKI, Stefan (Eds.): *Evaluation of Novel Approaches to Software Engineering* Bd. 69. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–14818–7, pp. 206–219

[ATKINSON AND TUNJIC 2014] ATKINSON, Colin; TUNJIC, Christian: Criteria for Orthographic Viewpoints. In: *Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. New York, NY, USA : ACM, 2014 (VAO '14). – ISBN 978–1–4503–2900–2, pp. 43:43–43:50

[ATKINSON ET AL. 2013b] ATKINSON, Colin; TUNJIC, Christian; STOLL, Dietmar ; ROBIN, Jacques: A Prototype Implementation of an Orthographic Software Modeling Environment. In: *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. New York, NY, USA : ACM, 2013 (VAO '13). – ISBN 978–1–4503–2070–2, pp. 3:1–3:10

[AVISON AND WOOD-HARPER 1991] AVISON, D. E.; WOOD-HARPER, A. T.: Information Systems Development Research: An Exploration of Ideas in Practice. In: *The Computer Journal* 34 (1991), No. 2, pp. 98–112

[AVISON AND WOOD-HARPER 1990] AVISON, David E.; WOOD-HARPER, A. T.: *Multiview: An Exploration in Information Systems Development*. Maidenhead : Alfred Waller Ltd., 1990. – ISBN 978–0632030262

[AVISON ET AL. 1998] AVISON, David E.; WOOD-HARPER, A. T.; VIDGEN, Richard T. ; WOOD, J. R. G.: A further exploration into information systems development: the evolution of Multiview2. In: *Information Technology & People* 11 (1998), No. 2, pp. 124–139

[BALZER 1991] BALZER, Robert: Tolerating Inconsistency. In: BELADY, Les (Eds.); BARSTOW, David R. (Eds.) ; TORII, Koji (Eds.): *Proceedings of the 13th International Conference on Software Engineering*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1991 (ICSE '91). – ISBN 0–89791–391–4, pp. 158–165

[BANCILHON AND SPYRATOS 1981] BANCILHON, François M.; SPYRATOS, Nicolas: Update Semantics of Relational Views. In: *ACM Transactions on Database Systems* 6 (1981), No. 4, pp. 557–575

[BARBERO ET AL. 2007] BARBERO, Mikael; DIDONET, Marcos; FABRO, Del ; BÉZIVIN, Jean: Traceability and provenance issues in global model management. In: *European Conference on Model Driven Architecture - Traceability Workshop*, 2007, pp. 47–55

[BERGMANN AND WILKE 1996] BERGMANN, Ralph; WILKE, Wolfgang: On the role of abstraction in case-based reasoning. In: SMITH, Ian (Eds.); FALTINGS, Boi (Eds.): *Advances in Case-Based Reasoning* Bd. 1168. Springer Berlin Heidelberg, 1996. – ISBN 978–3–540–61955–0, pp. 28–43

[BÉZIVIN 2005] BÉZIVIN, Jean: On the unification power of models. In: *Software & Systems Modeling* 4 (2005), No. 2, pp. 171–188

[BÉZIVIN ET AL. 2003] BÉZIVIN, Jean; GÉRARD, Sébastien; MULLER, Pierre-Alain ; RIOUX, Laurent: MDA Components: Challenges and Opportunities. In: EVANS, Andy (Eds.); SAMMUT, Paul (Eds.) ; WILLANS, James S. (Eds.): *Workshop on Metamodelling for MDA*. York, 2003, pp. 23–41

[BÉZIVIN AND GERBÉ 2001] BÉZIVIN, Jean; GERBÉ, Olivier: Towards a Precise Definition of the OMG/MDA Framework. In: *Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001)* IEEE, 2001, pp. 273–280

[BHAVE ET AL. 2011] BHAVE, Ajinkya; KROGH, Bruce H.; GARLAN, David ; SCHMERL, Bradley: View Consistency in Architectures for Cyber-Physical Systems. In: *IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS)*, 2011, pp. 151–160

[BICHLER 2014] BICHLER, Martin: Reflections on the State of Design Science Research. In: *Business & Information Systems Engineering* 6 (2014), No. 2, pp. 71–72

[BLAAUW AND BROOKS JR. 1997] BLAAUW, Gerrit A.; BROOKS JR., Frederick P.: *Computer Architecture: Concepts and Evolution*. Addison-Wesley Longman Publishing Co., Inc., 1997. – ISBN 978–0201105575

[BOBRIK ET AL. 2007] BOBRIK, Ralph; REICHERT, Manfred ; BAUER, Thomas: View-Based Process Visualization. In: ALONSO, Gustavo (Eds.); DADAM, Peter (Eds.) ; ROSEMANN, Michael (Eds.): *Business Process Management* Bd. 4714. Springer Berlin Heidelberg, 2007. – ISBN 978–3–540–75182–3, pp. 88–95

[BOCK ET AL. 2014] BOCK, Alexander; KACZMAREK, Monika; OVERBEEK, Sietse ; HESS, Michael: A Comparative Analysis of Selected Enterprise Modeling Approaches. In: FRANK, Ulrich (Eds.); LOUCOPOULOS, Pericles (Eds.); PASTOR, Oscar (Eds.) ; PETROUNIAS, Ilias (Eds.): *The Practice of Enterprise Modeling* Bd. 197. Springer Berlin Heidelberg, 2014, pp. 148–163

[BOOCH 1982] BOOCH, Grady: Object-Oriented Design. In: *Ada Lett.* I (1982), März, No. 3, pp. 64–76. – ISSN 1094–3641

[BOOCH 1991] BOOCH, Grady: *Object Oriented Design with Applications*. Redwood City, CA, USA : Benjamin-Cummings Publishing Co., Inc., 1991. – ISBN 0–8053–0091–0

[BÖRGER 2012] BÖRGER, Egon: Approaches to Modeling Business Processes. A Critical Analysis of BPMN, Workflow Patterns and YAWL. In: *Software and Systems Modeling* 11 (2012), No. 3, pp. 305–318

[BORK AND FILL 2014] BORK, Domenik; FILL, Hans-Georg: Formal Aspects of Enterprise Modeling Methods: A Comparison Framework. In: SPRAGUE, Ralph H. J. (Eds.): *Proceedings of the 47th Hawaii International Conference on System Sciences*. Big Island, Hawaii, USA : IEEE Computer Society Press, 2014 (HICSS'2014), pp. 3400–3409

[BORK AND KARAGIANNIS 2014] BORK, Domenik; KARAGIANNIS, Dimitris: Model-Driven Development of Multi-View Modeling Tools: The MUVIEMOT Approach. In: HOLZINGER, Andreas (Eds.); CARDOSO, Jorge (Eds.); CORDEIRO, José (Eds.); SINDEREN, Marten van (Eds.) ; MELLOR, Stephen (Eds.): *Proceedings of the 9th International Conference Software Paradigm Trends*, SCITEPRESS - Science and Technology Publications, 2014 (ICSOFT-PT'2014), pp. 11–23

[BORK AND SINZ 2010] BORK, Domenik; SINZ, Elmar J.: Design of a SOM Business Process Modelling Tool based on the ADOxx Meta-modelling Platform. In: LARA, Juan de (Eds.); VARRO, Daniel (Eds.); MARGARIA, Tiziana (Eds.); PADBERG, Julia (Eds.) ; TAENTZER, Gabriele (Eds.): *Pre-Proceedings of the 4th International Workshop on Graph-Based Tools (GraBaTs 2010), University of Twente, Enschede, The Netherlands*, 2010, pp. 90–101

[BORK AND SINZ 2011a] BORK, Domenik; SINZ, Elmar J.: Ein Multi-View-Modellierungswerkzeug für SOM-Geschäftsprozessmodelle auf Basis der Meta-Modellierungsplattform ADOxx. In: SINZ, Elmar J. (Eds.); BARTMANN, Dieter (Eds.); BODENDORF, Freimut (Eds.) ; FERSTL, Otto K. (Eds.): *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse* Bd. 9. Bamberg : University of Bamberg Press, 2011. – ISBN 9783863090098, pp. 369–384

[BORK AND SINZ 2011b] BORK, Domenik; SINZ, Elmar J.: *Modellierungsszenario, Anwendungsfall und konzeptueller Entwurf eines Multi-View-Modellierungswerkzeugs für das Semantische Objektmodell, 12th forFLEX Plenary Meeting, 2011-05-02, Nuremberg*. 2011

[BORK AND SINZ 2013] BORK, Domenik; SINZ, Elmar J.: Bridging the Gap from a Multi-View Modelling Method to the Design of a Multi-View Modeling Tool. In: *Enterprise Modelling and Information Systems Architectures (EMISA) - An International Journal* 8 (2013), No. 2, pp. 25–41

[BOUCKÉ ET AL. 2008] BOUCKÉ, Nelis; WEYNS, Danny; HILLIARD, Rich; HOLVOET, Tom ; HELLEBOOGH, Alexander: Characterizing Relations between Architectural Views. In: MORRISON, Ron (Eds.); BALASUBRAMANIAM, Dharini (Eds.) ; FALKNER, Katrina (Eds.): *Software Architecture* Bd. 5292. Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–88029–5, pp. 66–81

[BOUCKÉ ET AL. 2010] BOUCKÉ, Nelis; WEYNS, Danny ; HOLVOET, Tom: Composition of architectural models: Empirical analysis and language support. In: *Journal of Systems and Software* 83 (2010), No. 11, pp. 2108–2127

[BOULANGER ET AL. 2010] BOULANGER, Frédéric; JACQUET, Christophe; HARDEBOLLE, Cécile ; ROUIS, Elyes: Modeling Heterogeneous Points of View with ModHel'X. In: GHOSH, Sudipto (Eds.): *Models in Software Engineering* Bd. 6002. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–12260–6, pp. 310–324

[BRAATZ AND BRANDT 2011] BRAATZ, Benjamin; BRANDT, Christoph: Rule-Based Integration of Domain-Specific Modelling Languages. In: *Electronic Communications of the EASST* 42 (2011)

[BRAATZ AND BRANDT 2014] BRAATZ, Benjamin; BRANDT, Christoph: A Framework for Families of Domain-specific Modelling Languages. In: *Software Systems Modeling* 13 (2014), No. 1, pp. 109–132. – ISSN 1619–1366

[BREU ET AL. 1997] BREU, Ruth; HINKEL, Ursula; HOFMANN, Christoph; KLEIN, Cornel; PAECH, Barbara; RUMPE, Bernhard ; THURNER, Veronika: Towards a Formalization of the Unified Modeling Language. In: AKŞIT, Mehmet (Eds.); MATSUOKA, Satoshi (Eds.): *Proceedings of the 1lth European Conference Object-Oriented Programming*. Springer Berlin Heidelberg, 1997 (ECOOP'97). – ISBN 978–3–540–63089–0, pp. 344–366

[BRINKKEMPER 1996] BRINKKEMPER, Sjaak: Method Engineering: Engineering of Information Systems Development Methods and Tools. In: *Information and Software Technology* 38 (1996), No. 4, pp. 275–280

[BROMAN ET AL. 2012] BROMAN, David; LEE, Edward A.; TRIPAKIS, Stavros ; TORNGREN, Martin: Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems. In: *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling* Innsbruck, Austria, 2012

[BROOKS ET AL. 2008] BROOKS, Christopher; CHENG, Chihhong; FENG, Thomas H.; LEE, Edward A. ; HANXLEDEN, Reinhard von: Model Engineering using Multimodeling. In: *1st International Workshop on Model Co-Evolution and Consistency Management (MCCM '08)*, 2008

[BROOKS JR 1995] BROOKS JR, Frederick P.: *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, 1995. – ISBN 978–0201835953

[BROY 2003] BROY, Manfred: Multi-view Modeling of Software Systems. In: AICHERNIG, Bernhard K. (Eds.); MAIBAUM, Tom (Eds.): *Formal Methods at the Crossroads. From Panacea to Foundational Support* Bd. 2757. Springer Berlin Heidelberg, 2003. – ISBN 978–3–540–20527–2, pp. 207–225

[BROY 2012] BROY, Manfred: Software and System Modeling: Structured Multi-view Modeling, Specification, Design and Implementation. In: HINCHEY, Mike (Eds.); COYLE, Lorcan (Eds.): *Conquering Complexity*. Springer London, 2012, pp. 309–372

[BROY ET AL. 2010] BROY, Manfred; FEILKAS, Martin; HERRMANNSDOERFER, Markus; MERENDA, Stefano ; RATIU, Daniel: Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments. In: *Proceedings of the IEEE* 98 (2010), No. 4, pp. 526–545

[BUNGE 1977] BUNGE, Mario: *Treatise on basic philosophy: Ontology I: the furniture of the world*. Bd. 1. Springer, 1977. – ISBN 978–9027707857

[BURGER 2013] BURGER, Erik: Flexible Views for Rapid Model-driven Development. In: *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. New York, NY, USA : ACM, 2013 (VAO '13). – ISBN 978–1–4503–2070–2, pp. 1:1–1:5

[CAO ET AL. 2005] CAO, Shuping; GRUNDY, John; HOSKING, John; STOECKLE, Hermann; TEMPERO, Ewan ; ZHU, Nianping: Generating Web-based User Interfaces for Diagramming Tools. In: *Proceedings of the Sixth Australasian Conference on User Interface - Volume 40*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2005 (AUIC '05), pp. 63–72

[CARLSON AND ARORA 1979] CARLSON, C. R.; ARORA, Adarsh K.: The updatability of relational views based on functional dependencies. In: *Computer Software and Applications Conference, Proceedings. COMPSAC 79. The IEEE Computer Society's Third International*, 1979, pp. 415–420

[CHEN AND SCHEER 1994] CHEN, Rong; SCHEER, August-Wilhelm: *Modellierung von Prozessketten mittels Petri-Netz-Theorie*. Universität des Saarlandes, Institut für Wirtschaftsinformatik, 1994 (Institut für Wirtschaftsinformatik Saarbrücken: Veröffentlichungen des Instituts für Wirtschaftsinformatik - Heft 107)

[CICCHETTI ET AL. 2011] CICCHETTI, Antonio; CICCOZZI, Federico ; LEVEQUE, Thomas: A hybrid approach for multi-view modeling. In: *Electronic Communication of the ECEASST* 50 (2011), pp. 368–377

[CICCHETTI ET AL. 2012] CICCHETTI, Antonio; CICCOZZI, Federico ; LEVEQUE, Thomas: Supporting Incremental Synchronization in Hybrid Multi-view Modelling. In: KIENZLE, Jörg (Eds.): *Models in Software Engineering* Bd. 7167. Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–29644–4, pp. 89–103

[CICCHETTI ET AL. 2007] CICCHETTI, Antonio; RUSCIO, Davide D. ; PIERANTONIO, Alfonso: A Metamodel Independent Approach to Difference Representation. In: *Journal of Object Technology* 6 (2007), No. 9, pp. 165–185. – ISSN 1660–1769

[COUNCIL 2012] COUNCIL, Supply C.: *Supply chain operations reference model*. Supply Chain Council, 2012

[CZARNECKI AND HELSEN 2003] CZARNECKI, Krzysztof; HELSEN, Simon: Classification of Model Transformation Approaches. In: *2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*. Anaheim, CA, USA, 2003

[DACONTA ET AL. 2003] DACONTA, Michael C.; OBRST, Leo J. ; SMITH, Kevin T.: *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing Inc., Indianapolis, Indiana, 2003. – ISBN 978–0471432579

[DAVISON ET AL. 2004] DAVISON, Robert; MARTINSONS, Maris G. ; KOCK, Ned: Principles of Canonical Action Research. In: *Information Systems Journal* 14 (2004), No. 1, pp. 65–86. – ISSN 1365–2575

[DAYAL AND BERNSTEIN 1978] DAYAL, Umeshwar; BERNSTEIN, Philip A.: On the Updatability of Relational Views. In: YAO, S. B. (Eds.): *VLDB*, IEEE Computer Society, 1978, pp. 368–377

[DAYAL AND BERNSTEIN 1982] DAYAL, Umeshwar; BERNSTEIN, Philip A.: On the Correct Translation of Update Operations on Relational Views. In: *ACM Transactions on Database Systems* 7 (1982), No. 3, pp. 381–416

[DAYAL AND HWANG 1984] DAYAL, Umeshwar; HWANG, Hai-Yann: View Definition and Generalization for Database Integration in a Multidatabase System. In: *IEEE Transactions on Software Engineering* 10 (1984), No. 6, pp. 628–645

[DEMATHIEU ET AL. 2005] DEMATHIEU, Sebastien; GRIFFIN, Catherine ; SENDALL, Shane: Model Transformation with the IBM Model Transformation Framework. In: *IBM alphaWorks* (2005). – `http://www.ibm.com/developerworks/rational/library/05/503_sebas/` last checked: 2015-04-16

[DEMOLY ET AL. 2010] DEMOLY, Frédéric; MONTICOLO, Davy; EYNARD, Benoit; RIVEST, Louis ; GOMES, Samuel: Multiple viewpoint modelling framework enabling integrated product-process design. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 4 (2010), No. 4, pp. 269–280. – ISSN 1955–2513

[DENNING 1997] DENNING, Peter J.: A New Social Contract for Research. In: *Communications of the ACM* 40 (1997), No. 2, pp. 132–134

[DIJKMAN 2006] DIJKMAN, Remco M.: *Consistency in Multi-Viewpoint Architectural Design*, University of Twente, Enschede, The Netherlands, PhD thesis, 2006

[DIJKMAN ET AL. 2006] DIJKMAN, Remco M.; QUARTEL, Dick A. C. ; SINDEREN, Marten J. V.: Consistency in Multi-Viewpoint Architectural Design of Enterprise Information systems / Eindhoven University of Technology, Eindhoven, The Netherlands. 2006. – Forschungsbericht. – Working Paper WP-188

[DIJKMAN ET AL. 2008] DIJKMAN, Remco M.; QUARTEL, Dick A. C. ; SINDEREN, Marten J.: Consistency in multi-viewpoint design of enterprise information systems. In: *Information Software Technology* 50 (2008), No. 7-8, pp. 737–752

[EASTERBROOK ET AL. 1994] EASTERBROOK, Steve; FINKELSTEIN, Anthony; KRAMER, Jeff ; NUSEIBEH, Bashar: Coordinating Distributed ViewPoints: the anatomy of a consistency check. In: *Concurrent Engineering* 2 (1994), No. 3, pp. 209–222

[EHRIG ET AL. 2007] EHRIG, Hartmut; EHRIG, Karsten; ERMEL, Claudia; HERMANN, Frank ; TAENTZER, Gabriele: Information Preserving Bidirectional Model Transformations. In: DWYER, Matthew B. (Eds.); LOPES, Antónia (Eds.): *Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering*. Berlin, Heidelberg : Springer-Verlag, 2007 (FASE'07). – ISBN 978–3–540–71288–6, pp. 72–86

[ELLSON ET AL. 2002] ELLSON, John; GANSNER, Emden; KOUTSOFIOS, Lefteris; NORTH, Stephen C. ; WOODHULL, Gordon: Graphviz - Open Source Graph Drawing Tools. In: MUTZEL, Petra (Eds.); JÜNGER, Michael (Eds.) ; LEIPERT, Sebastian (Eds.): *Graph Drawing* Bd. 2265. Springer Berlin Heidelberg, 2002. – ISBN 978–3–540–43309–5, pp. 483–484

[ELLSON ET AL. 2004] ELLSON, John; GANSNER, Emden R.; KOUTSOFIOS, Eleftherios; NORTH, Stephen C. ; WOODHULL, Gordon: Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools. In: JÜNGER, Michael (Eds.); MUTZEL, Petra (Eds.): *Graph Drawing Software*. Springer Berlin Heidelberg, 2004 (Mathematics and Visualization). – ISBN 978–3–642–62214–4, pp. 127–148

[EMERSON AND SZTIPANOVITS 2006] EMERSON, Matthew; SZTIPANOVITS, Janos: Techniques for Metamodel Composition. In: *Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling, OOPSLA 2006*, Association for Computing Machinery (ACM), ACM Press, 2006, pp. 123–139

[EMERY AND HILLIARD 2008] EMERY, David; HILLIARD, Rich: Updating IEEE 1471: Architecture Frameworks and Other Topics. In: *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on Software Architecture*, 2008, pp. 303–306

[ENGELS ET AL. 2001] ENGELS, Gregor; KÜSTER, Jochem M.; HECKEL, Reiko ; GROENEWEGEN, Luuk: A Methodology for Specifying and Analyzing Consistency of Object-Oriented Behavioral Models. In: *Proceedings of the 8th European Software Engineering Conference*, Association for Computing Machinery (ACM), 2001 (ESEC'2001). – ISBN 1–58113–390–1, pp. 186–195

[ERAMO ET AL. 2008] ERAMO, Romina; PIERANTONIO, Alfonso; ROMERO, José R. ; VALLECILLO, Antonio: Change Management in Multi-Viewpoint Systems using ASP. In: SINDEREN, Marten van (Eds.); ALMEIDA, Joao Paulo A. (Eds.); PIRES, Luís F. (Eds.) ; STEEN, Maarten (Eds.): *12th Enterprise Distributed Object Computing Conference Workshops*, 2008, pp. 433–440

[ESPRIT CONSORTIUM AMICE 1998] ESPRIT CONSORTIUM AMICE, The: *CIMOSA: Open System Architecture for CIM*. 2. Springer-Verlag, Berlin, 1998. – ISBN 978–3540562566

[EVERMANN AND WAND 2001] EVERMANN, Joerg; WAND, Yair: Towards Ontologically Based Semantics for UML Constructs. In: KUNII, Hideko S. (Eds.); JAJODIA, Sushil (Eds.) ; SØLVBERG, Arne (Eds.): *Proceedings of the 20th International Conference on Conceptual Modeling*. Springer Berlin Heidelberg, 2001 (ER'01). – ISBN 978–3–540–42866–4, pp. 354–367

[FAVRE 2004] FAVRE, Jean-Marie: Foundations of Model (Driven) (Reverse) Engineering: Models – Episode I: Stories of the Fidus Papyrus and of the Solarus. In: *Post-Proceedings of Dagstuhl Seminar on Model Driven Reverse Engineering*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2004, pp. 1–31

[FAVRE 2005] FAVRE, Jean-Marie: Foundations of Meta-Pyramids: Languages vs. Metamodels-Episode II: Story of Thotus the Baboon. In: *Language Engineering for Model-Driven Software Development* 4101 (2005)

[FERSTL AND SINZ 1990] FERSTL, Otto K.; SINZ, Elmar J.: Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). In: *Wirtschaftsinformatik* 32 (1990), No. 6, pp. 566–581

[FERSTL AND SINZ 1991] FERSTL, Otto K.; SINZ, Elmar J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). In: *Wirtschaftsinformatik* 33 (1991), No. 6, pp. 477–491

[FERSTL AND SINZ 1995] FERSTL, Otto K.; SINZ, Elmar J.: Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. In: *Wirtschaftsinformatik* 37 (1995), No. 3, pp. 209–220

[FERSTL AND SINZ 1996] FERSTL, Otto K.; SINZ, Elmar J.: Geschäftsprozessmodellierung im Rahmen des Semantischen Objektmodells. In: VOSSEN, Gottfried (Eds.); BECKER, Jörg (Eds.): *Geschäftsprozessmodellierung und Workflow-Management*. Bonn, International Thomson Publisher, 1996, pp. 47–62

[FERSTL AND SINZ 2006] FERSTL, Otto K.; SINZ, Elmar J.: Modeling of Business Systems Using SOM. In: BERNUS, Peter (Eds.); MERTINS, Kai (Eds.) ; SCHMIDT, Günter (Eds.): *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, 2006 (International Handbooks on Information Systems). – ISBN 978–3–540–25472–0, pp. 347–367

[FERSTL AND SINZ 2008] FERSTL, Otto K.; SINZ, Elmar J.: *Grundlagen der Wirtschaftsinformatik*. 6th. München : Oldenbourg, 2008. – ISBN 9783486719178

[FERSTL AND SINZ 2013] FERSTL, Otto K.; SINZ, Elmar J.: *Grundlagen der Wirtschaftsinformatik*. 7th. München : Oldenbourg, 2013. – ISBN 9783486719178

[FETTKE AND LOOS 2003] FETTKE, Peter; LOOS, Peter: Model Driven Architecture (MDA). In: *Wirtschaftsinformatik* 45 (2003), No. 5, pp. 555–559

[FILL 2005] FILL, Hans-Georg: UML Statechart Diagrams on the ADONIS Metamodeling Platform. In: *Electronic Notes in Theoretical Computer Science* 127 (2005), No. 1, pp. 27–36

[FILL 2012] FILL, Hans-Georg: SeMFIS: A Tool for Managing Semantic Conceptual Models. In: STÖRRLE, Harald (Eds.): *ECMFA 2012 - Joint Proceedings Co-located Events at the 8th European Conference on Modelling Foundations and Applications, Technical University of Denmark*. Lyngby, Denmark, 2012, pp. 229–240

[FILL ET AL. 2013] FILL, Hans-Georg; HICKL, Susan; KARAGIANNIS, Dimitris; OBERWEIS, Andreas ; SCHOKNECHT, Andreas: A Formal Specification of the Horus Modeling Language Using FDMM. In: ALT, Rainer (Eds.); FRANCZYK, Bogdan (Eds.): *Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013)*, Merkur-Verlag, 2013, pp. 1165 – 1179

[FILL AND KARAGIANNIS 2013] FILL, Hans-Georg; KARAGIANNIS, Dimitris: On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. In: *Enterprise Modelling and Information Systems Architectures (EMISA) - An International Journal* 8 (2013), No. 1, pp. 4–25

[FILL ET AL. 2012] FILL, Hans-Georg; REDMOND, Tim ; KARAGIANNIS, Dimitris: FDMM: A Formalism for Describing ADOxx Meta Models and Models. In: MACIASZEK, Leszek (Eds.); CUZZOCREA, Alfredo (Eds.) ; CORDEIRO, José (Eds.): *ICEIS 2012 - 14th International Conference on Enterprise Information Systems*, 2012, pp. 133–144

[FINKELSTEIN ET AL. 1992] FINKELSTEIN, A.; KRAMER, J.; NUSEIBEH, B.; FINKELSTEIN, L. ; GOEDICKE, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. In: *International Journal of Software Engineering and Knowledge Engineering* 2 (1992), pp. 31–57

[FINKELSTEIN AND FUKS 1989] FINKELSTEIN, Anthony; FUKS, Hugo: Multiparty Specification. In: *Proceedings of the 5th International Workshop on Software Specification and Design*. New York, NY, USA : ACM, 1989 (IWSSD '89). – ISBN 0–89791–305–1, pp. 185–195

[FINKELSTEIN ET AL. 1993] FINKELSTEIN, Anthony; GABBAY, Dov; HUNTER, Anthony; KRAMER, Jeff ; NUSEIBEH, Bashar: Inconsistency Handling in Multi-Perspective Specifications. In: SOMMERVILLE, Ian (Eds.); PAUL, Manfred (Eds.): *Software Engineering - ESEC '93* Bd. 717. Springer Berlin Heidelberg, 1993. – ISBN 978–3–540–57209–1, pp. 84–99

[FINKELSTEIN ET AL. 1991] FINKELSTEIN, Anthony; GOEDICKE, Michael; KRAMER, Jeff ; NISKIER, Celso: ViewPoint oriented software development: Methods and viewpoints in requirements engineering. In: BERGSTRA, J.A. (Eds.); FEIJS, L.M.G. (Eds.): *Algebraic Methods II: Theory, Tools and Applications* Bd. 490. Springer Berlin Heidelberg, 1991. – ISBN 978–3–540–53912–4, pp. 29–54

[FISCHER ET AL. 2000] FISCHER, Thorsten; NIERE, Jörg; TORUNSKI, Lars ; ZÜNDORF, Albert: Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In: EHRIG, Hartmut (Eds.); ENGELS, Gregor (Eds.); KREOWSKI, Hans-Jörg (Eds.) ; ROZENBERG, Grzegorz (Eds.): *Theory and Application of Graph Transformations* Bd. 1764. Springer Berlin Heidelberg, 2000. – ISBN 978–3–540–67203–6, pp. 296–309

[FONTOURA ET AL. 2001] FONTOURA, Marcus; PREE, Wolfgang ; RUMPE, Bernhard: *The UML profile for framework architectures*. New York : Addison-Wesley, 2001 (Addison-Wesley object technology series). – ISBN 978–0201675184

[FOX 1992] FOX, Mark S.: The TOVE Project Towards a Common-Sense Model of the Enterprise. In: BELLI, Fevzi (Eds.); RADERMACHER, Franz J. (Eds.): *Proceedings of the 5th international conference on Industrial and engineering applications of artificial intelligence and expert systems*. London, UK, UK : Springer-Verlag, 1992 (IEA/AIE '92). – ISBN 3–540–55601–X, pp. 25–34

[FOX ET AL. 1993] FOX, Mark S.; CHIONGLO, John F. ; FADEL, Fadi G.: A Common-Sense Model of the Enterprise. In: MITTA, Deborah A. (Eds.): *Proceedings of the Industrial Engineering Research Conference*, Institute of Industrial Engineers, 1993, pp. 425–429

[FOX AND GRUNINGER 1998] FOX, Mark S.; GRUNINGER, Michael: Enterprise Modeling. In: *American Association for Artificial Intelligence* 19 (1998), No. 3, pp. 109–122

[FRANCE ET AL. 1998] FRANCE, Robert; EVANS, Andy; LANO, Kevin ; RUMPE, Bernhard: The UML as a Formal Modeling Notation. In: *Computer Standards & Interfaces* 19 (1998), No. 7, pp. 325 – 334

[FRANCE AND RUMPE 2007] FRANCE, Robert; RUMPE, Bernhard: Model-driven Development of Complex Software: A Research Roadmap. In: *2007 Future of Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2007 (FOSE '07). – ISBN 0–7695–2829–5, pp. 37–54

[FRANK 1992] FRANK, Ulrich: A Tool Supported Methodology for Designing Multi-Perspective Enterprise Models. In: MCGREGOR, R. (Eds.): *Proceedings of the Third Australian Conference on Information Systems*. 1992, pp. 675–690

[FRANK 1994] FRANK, Ulrich: MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems. In: EGE, Raimund (Eds.); SINGH, Madhu (Eds.) ; MEYER, Bertrand (Eds.): *Technology of Object-Oriented Languages ans Systems*. 1994, pp. 367–380

[FRANK 1999] FRANK, Ulrich: Eine Architektur zur Spezifikation von Sprachen und Werkzeugen für die Unternehmensmodellierung. In: SINZ, Elmar J. (Eds.): *Proceedings of the Symposium on Modellierung betrieblicher Informationssysteme, Bamberg*, 1999 (MobIS), pp. 154–169

[FRANK 2002] FRANK, Ulrich: Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. Washington, DC, USA : IEEE Computer Society, 2002 (HICSS '02), pp. 72–82

[FRANK 2010] FRANK, Ulrich: The MEMO Meta Modelling Language (MML) and Language Architecture / University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB). 2010 (24). – ICB Research Reports

[FRANK 2011a] FRANK, Ulrich: MEMO Organisation Modelling Language (1):Focus on Organizational Structure / University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB). 2011 (48). – ICB Research Reports

[FRANK 2011b] FRANK, Ulrich: MEMO Organisation Modelling Language (2):Focus on Business Processes / University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB). 2011 (49). – ICB Research Reports

[FRANK 2011c] FRANK, Ulrich: MEMO Organisation Modelling Language: Requirements and core diagram types / University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB). 2011 (47). – ICB Research Reports

[FRANK 2011d] FRANK, Ulrich: Some Guidelines for the Conception of Domain-Specific Modelling Languages. In: NÜTTGENS, M. (Eds.); THOMAS, O. (Eds.) ; WEBER, B. (Eds.): *Proceedings of the Enterprise Modeling and Information Systems Architecture Workshop*, 2011, pp. 93–106

[FRANK 2012] FRANK, Ulrich: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. In: *Software & Systems Modeling* 13 (2012), No. 3, pp. 1–22. – ISSN 1619–1366

[FRANK ET AL. 2014] FRANK, Ulrich; STRECKER, Stefan; FETTKE, Peter; BROCKE, Jan vom; BECKER, Jörg ; SINZ, Elmar J.: The Research Field "Modeling Business Information Systems". In: *Business & Information Systems Engineering* 6 (2014), No. 1, pp. 1–5

[FUENTES ET AL. 2002] FUENTES, Lidia; TROYA, José M.; VALLECILLO, Antonio ; VALLECILLO, Antonio: Using UML Profiles for Documenting Web-Based Application Frameworks. In: *Annals of Software Engineering* 13 (2002), pp. 249–264

[FUENTES-FERNÁNDEZ AND VALLECILLO-MORENO 2004] FUENTES-FERNÁNDEZ, Lidia; VALLECILLO-MORENO, Antonio: An Intorduction to UML Profiles. In: *The Europpean Jorunal for the Informatics Professional, UML and Model Engineering* 5 (2004), No. 2, pp. 1–13

[FUHRMANN AND VON HANXLEDEN 2010] FUHRMANN, Hauke; HANXLEDEN, Reinhard von: Taming Graphical Modeling. In: PETRIU, Dorina C. (Eds.); ROUQUETTE, Nicolas (Eds.) ; HAUGEN, Øystein (Eds.): *Model Driven Engineering Languages and Systems* Bd. 6394. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–16144–5, pp. 196–210

[FUHRMANN 2011] FUHRMANN, Hauke A. L.: *On the Pragmatics of Graphical Modeling*, Kiel University, PhD thesis, 2011

[GANSNER ET AL. 2010] GANSNER, Emden R.; KOUTSOFIOS, Eleftherios ; NORTH, Stephen: Drawing graphs with dot / Technical Report, AT&T Research. 2010. – Forschungsbericht. – [Online] `http://bsdwatch.net/docs/userdocs/graphviz/pdf/dotguide.pdf`, last checked: 2015-03-28

[GEISLER ET AL. 1998] GEISLER, Robert; KLAR, Marcus ; PONS, Claudia: Dimensions and Dichotomy in Metamodeling. Technical Report / Technical University of Berlin, Germany. 1998. – Forschungsbericht

[GIESE AND WAGNER 2009] GIESE, Holger; WAGNER, Robert: From model transformation to incremental bidirectional model synchronization. In: *Software & Systems Modeling* 8 (2009), No. 1, pp. 21–43

[GOLDSCHMIDT ET AL. 2012] GOLDSCHMIDT, Thomas; BECKER, Steffen ; BURGER, Erik: Towards a Tool-Oriented Taxonomy of View-Based Modelling. In: SINZ, Elmar J. (Eds.); SCHÜRR, Andy (Eds.): *Modellierung 2012, Bamberg, Deutschland*, Gesellschaft für Informatik (GI), 2012 (Lecture Notes in Informatics), pp. 59–74

[GREGOR AND HEVNER 2013] GREGOR, Shirley; HEVNER, Alan R.: Positioning and Presenting Design Science Research for Maximum Impact. In: *MIS Quarterly* 37 (2013), No. 2, pp. 337–356. – ISSN 0276–7783

[GRUHN ET AL. 2008] GRUHN, Volker; LAUE, Ralf; KERN, Heiko ; KÜHNE, Stefan: EPK-Validierung zur Modellierungszeit in der bflow* Toolbox. In: LOOS, Peter (Eds.); NÜTTGENS, Markus (Eds.); TUROWSKI, Klaus (Eds.) ; WERTH, Dirk (Eds.): *Modellierung betrieblicher Informationssysteme (MobIS 2008)*, Bonner Köllen Verlag, 2008 (Lecture Notes in Informatics), pp. 181–194

[GRUNDY AND HOSKING 1993] GRUNDY, John; HOSKING, John: The MViews framework for constructing multi-view editing environments. In: *New Zealand Journal of Computing* 4 (1993), No. 2, pp. 31–40

[GRUNDY AND HOSKING 2007] GRUNDY, John; HOSKING, John: Supporting Generic Sketching-Based Input of Diagrams in a Domain-Specific Visual Language Meta-Tool. In: *Proceedings of the 29th International Conference on Software Engineering*, 2007 (ICSE 2007). – ISSN 0270–5257, pp. 282–291

[GRUNDY ET AL. 1991] GRUNDY, John; HOSKING, John ; HAMER, John: A Visual Programming Environment for Object-Oriented Languages. In: KORSON, Timothy D. (Eds.); VASHNAVI, Vijay (Eds.) ; MEYER, Bertrand (Eds.): *Proceedings TOOLS 1991: 5th International Conference on Technology of Object-Oriented Languages and Systems*, Prentice Hall, 1991, pp. 129–138

[GRUNDY ET AL. 2008] GRUNDY, John; HOSKING, John; HUH, Jun ; LI, Karen: Marama: An Eclipse Meta-toolset for Generating Multi-view Environments. In: *Proceedings of the 30th International Conference on Software Engineering*. New York, NY, USA : ACM, 2008 (ICSE '08). – ISBN 978–1–60558–079–1, pp. 819–822

[GRUNDY ET AL. 2013] GRUNDY, John; HOSKING, John; LI, Karen N.; ALI, Norhayati M.; HUH, Jun ; LI, Richard L.: Generating Domain-Specific Visual Language Tools from Abstract Visual Specifications. In: *IEEE Transactions on Software Engineering* 39 (2013), No. 4, pp. 487–515. – ISSN 0098–5589

[GRUNDY AND HOSKING 1998] GRUNDY, John C.; HOSKING, John G.: Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination. In: NITTO, Elisabetta (Eds.); FUGGETTA, Alfonso (Eds.): *Process Technology*. Springer US, 1998. – ISBN 978–1–4613–7484–8, pp. 27–60

[GULDEN AND FRANK 2010] GULDEN, Jens; FRANK, Ulrich: MEMOCENTERNG: A full-featured modeling environment for organisation modeling and model-driven software development. In: SOFFER, Pnina (Eds.); PROPER, Erik (Eds.): *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*. Berlin, Heidelberg : Springer, 2010 (Lecture Notes in Business Information Processing), pp. 76–83

[GUPTA AND KUMAR 2014] GUPTA, Avadhesh K.; KUMAR, Satish: An Extensive Study on the Future of Modeling in Software Development. In: *International Journal on Computer Science and Engineering* 6 (2014), No. 8, pp. 300–305

[HAILPERN AND TARR 2006] HAILPERN, Brent; TARR, Peri L.: Model-driven Development: The Good, the Bad, and the Ugly. In: *IBM Systems Journal* 45 (2006), No. 3, pp. 451–461. – ISSN 0018–8670

[VON HANXLEDEN ET AL. 2012] HANXLEDEN, Reinhard von; LEE, Edward A.; MOTIKA, Christian ; FUHRMANN, Hauke: Multi-view Modeling and Pragmatics in 2020. In: CALINESCU, Radu (Eds.); GARLAN, David (Eds.): *Large-Scale Complex IT Systems. Development, Operation and Management* Bd. 7539. Springer Berlin Heidelberg, 2012, pp. 209–223

[HAPPEL AND SEEDORF 2006] HAPPEL, Hans-Jörg; SEEDORF, Stefan: Applications of ontologies in software engineering. In: *Proceedings of Workshop on Sematic Web Enabled Software Engineering (SWESE) on the ISWC* Citeseer, 2006, pp. 5–9

[HAREL AND RUMPE 2004] HAREL, David; RUMPE, Bernhard: Meaningful Modeling: What's the Semantics of "Semantics"? In: *Computer* 37 (2004), No. 10, pp. 64–72

[HARMON 2010] HARMON, Paul: *The BPTrends 2010 BPM Software Tools Report on BOC's Adonis Version 4.0*. 2010. – `http://www.bptrends.com/publicationfiles/2010%20BPM%20Tools%20Report-BOCph.pdf`, last checked: 2013-10-31

[HARTMANN 2011] HARTMANN, Beate: Enterprise Architecture as an Instrument of Strategic Control. In: NÜTTGENS, Markus (Eds.); THOMAS, Oliver (Eds.) ; WEBER, Barbara (Eds.): *Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011, Hamburg, Germany* Bd. 190, Gesellschaft für Informatik, 2011 (LNI), pp. 9–22

[HARTMANN 2015] HARTMANN, Beate: *Die Unternehmensarchitektur als Instrument der strategischen Kontrolle*, University of Bamberg, PhD thesis, 2015

[HARTMANN ET AL. 2013] HARTMANN, Beate; TEUSCH, Andree ; WOLF, Matthias: Spezifikation von funktionalen und nichtfunktionalen Systemanforderungen auf Basis von Geschäftsprozessmodellen. In: RAINER ALT AND BOGDAN FRANCZYK (Eds.): *Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013)*, 2013, pp. 1293–1307

[HARTMANN AND WOLF 2012] HARTMANN, Beate; WOLF, Matthias: Erweiterung einer Geschäftsprozessmodellierungssprache zur Stärkung der strategischen Ausrichtung von Geschäftsprozessen. In: SINZ, Elmar J. (Eds.); SCHÜRR, Andy (Eds.): *Proceedings Modellierung*, 2012, pp. 235–250

[HEBIG ET AL. 2011] HEBIG, Regina; SEIBEL, Andreas ; GIESE, Holger: On the Unification of Megamodels. In: *Electronic Communications of the EASST* 42 (2011)

[HEISE ET AL. 2014] HEISE, David; STRECKER, Stefan ; FRANK, Ulrich: CONTROLML: A domain-specific modeling language in support of assessing internal controls and the internal control system. In: *International Journal of Accounting Information Systems* 15 (2014), No. 3, pp. 224–245. – ISSN 1467–0895

[HERBST 2001] HERBST, Joachim: *Ein induktiver Anstaz zur Akquisition und Adaption von Workflow-Modellen*, University of Ulm, PhD thesis, 2001

[HERBST ET AL. 1997] HERBST, Joachim; JUNGINGER, Stefan ; KÜHN, Harald: Simulation in Financial Services with the Business Process Management System ADONIS. In: HAHN, Winfried (Eds.); LEHMANN, Axel (Eds.): *Proceedings of the 9th European Simulation Symposium (ESS'97)*, 1997

[HERZIG ET AL. 2014] HERZIG, Sebastian J.; QAMAR, Ahsan ; PAREDIS, Christiaan J.: An Approach to Identifying Inconsistencies in Model-based Systems Engineering. In: *Procedia Computer Science* 28 (2014), pp. 354–362

[HERZIG ET AL. 2011] HERZIG, Sebastian J.; QAMAR, Ahsan; REICHWEIN, Axel ; PAREDIS, Christian J.: A Conceptual Framework for Consistency Management in Model-Based Systems Engineering. In: *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2011, pp. 1329–1339

[HESSELLUND 2009] HESSELLUND, Anders: *Domain-Specific Multimodeling*, IT-University of Copenhagen, Denmark, PhD thesis, 2009

[HEVNER 2007] HEVNER, Alan R.: A Three Cycle View of Design Science Research. In: *Scandinavian Journal of Information Systems* 19 (2007), No. 2, pp. 87–92

[HEVNER ET AL. 2004] HEVNER, Alan R.; MARCH, Salvatore T.; PARK, Jinsoo ; RAM, Sudha: Design Science in Information Systems Research. In: *MIS Quaterly* 28 (2004), No. 1, pp. 75–105. – ISSN 0276–7783

[HILLIARD 1999] HILLIARD, Rich: Views and Viewpoints in Software Systems Architecture. In: DONOHOE, Patrick (Eds.): *First Working IFIP Conference on Software Architecture*, 1999, pp. 1–10

[HILLIARD 2012] HILLIARD, Rich: *Architecture Viewpoint Template for ISO/IEC/IEEE 42010, Version 2.1b*. 2012

[HÖFFERER 2007] HÖFFERER, Peter: Achieving business process model interoperability using metamodels and ontologies. In: *Proceedings of the 15th European Conference on Information Systems (ECIS 2007)*, 2007, pp. 1620–1631

[HOFFMANN ET AL. 1992] HOFFMANN, W.; KIRSCH, J. ; SCHEER, A.-W.: *Modellierung mit ereignisgesteuerten Prozeßketten: Methodenhandbuch*. Universität des Saarlandes, Institut für Wirtschaftsinformatik, 1992 (Institut für Wirtschaftsinformatik Saarbrücken: Veröffentlichungen des Instituts für Wirtschaftsinformatik - Heft 101)

[HOUY ET AL. 2012] HOUY, Constantin; FETTKE, Peter ; LOOS, Peter: Understanding Understandability of Conceptual Models - What Are We Actually Talking about? In: ATZENI, Paolo (Eds.); CHEUNG, David (Eds.) ; RAM, Sudha (Eds.): *Proceedings of the 2012 International Conference on Conceptual Modeling, ER 2012* Bd. 7532. Springer Berlin Heidelberg, 2012, pp. 64–77

[HUZAR ET AL. 2005] HUZAR, Zbigniew; KUZNIARZ, Ludwik; REGGIO, Gianna ; SOURROUILLE, JeanLouis: Consistency Problems in UML-Based Software Development. In: JARDIM NUNES, Nuno (Eds.); SELIC, Bran (Eds.); SILVA, Alberto Rodrigues d. (Eds.) ; TOVAL ALVAREZ, Ambrosio (Eds.): *Proceedings of the 2005 International Conference on UML Modeling Languages and Applications* Bd. 3297. Springer Verlag Berlin Heidelberg, 2005, pp. 1–12

[IEEE 2011] IEEE, ISO: Systems and software engineering – Architecture description. In: *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Standard 1471-2000)* (2011), pp. 1–46

[IIVARI 2007] IIVARI, Juhani: A Paradigmatic Analysis of Information Systems as a Design Science. In: *Scandinavian Journal of Information Systems* 19 (2007), No. 2, pp. 39–64

[JACKSON 1990] JACKSON, Michael: Some Complexities in Computer-based Systems and Their Implications for System Development. In: *Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering*, 1990, pp. 344–351

[JACOBSON 1992] JACOBSON, Ivar: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Longman, Amsterdam, 1992. – ISBN 978–0201544350

[JARRAYA AND DEBBABI 2012] JARRAYA, Yosr; DEBBABI, Mourad: Formal Specification and Probabilistic Verification of SysML Activity Diagrams. In: MARGARIA, Tiziana (Eds.); QIU, Zongyan (Eds.) ; YANG, Hongli (Eds.): *Sixth International Symposium on Theoretical Aspects of Software Engineering (TASE'2013)*, 2012, pp. 17–24

[JARRAYA ET AL. 2009] JARRAYA, Yosr; DEBBABI, Mourad ; BENTAHAR, Jamal: On the Meaning of SysML Activity Diagrams. In: *16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, IEEE Computer Society, 2009 (ECBS'09), pp. 95–105

[JUNG ET AL. 2004] JUNG, Moonyoung; KIM, Hak S.; JO, Myung H.; TAK, Kyung H.; CHA, Hyun S. ; SON, Jin H.: Mapping from BPMN-Formed Business Processes to XPDL Business Processes. In: CHEN, Jian (Eds.): *Proceedings of the International Conference on Electronic Business (ICEB'04)*, Academic Publishers/World Publishing Corporation, 2004, pp. 422–427

[JUNGINGER ET AL. 2000] JUNGINGER, Stefan; KÜHN, Harald; STROBL, Robert ; KARAGIANNIS, Dimitris: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. In: *Wirtschaftsinformatik* 42 (2000), No. 5, pp. 392–401

[KARAGIANNIS ET AL. 2008] KARAGIANNIS, Dimitris; GROSSMANN, Wilfried ; HOEFFERER, Peter: *Open Model Initiative: A Feasilbility Study*. 2008. – `http://cms.dke.univie.ac.at/uploads/media/Open\_Models\_Feasibility\_Study\_SEPT\_2008.pdf`, last checked: 2013-07-14

[KARAGIANNIS ET AL. 1996] KARAGIANNIS, Dimitris; JUNGINGER, Stefan ; STROBL, Robert: Introduction to Business Process Management Systems Concepts. In: SCHOLZ-REITER, Bernd (Eds.); STICKEL, Eberhard (Eds.): *Business Process Modelling*. Springer Berlin Heidelberg, 1996. – ISBN 978–3–642–80319–2, pp. 81–106

[KARAGIANNIS AND KÜHN 2002] KARAGIANNIS, Dimitris; KÜHN, Harald: Metamodeling Platforms. In: BAUKNECHT, K. (Eds.); MIN TJOA, A. (Eds.) ; QUIRCHMAYR, G. (Eds.): *Third International Conference EC-Web 2002 - Dexa 2002*. Aix-en-Provence, France : Springer-Verlag, Berlin, Heidelberg, 2002, pp. 182

[KARAGIANNIS AND SCHWAB 2013] KARAGIANNIS, Dimitris; SCHWAB, Margit: An Engineering Approach for the Design of Hybrid Modelling Methods. In: CORDEIRO, José (Eds.); MACIASZEK, Leszek A. (Eds.) ; FILIPE, Joaquim (Eds.): *Proceedings of the 14th International Conference on Enterprise Information Systems - ICEIS 2012* Bd. 141. Springer Berlin Heidelberg, 2013. – ISBN 978–3–642–40653–9, pp. 3–17

[KARAGIANNIS AND TELESKO 2000] KARAGIANNIS, Dimitris; TELESKO, Rainer: The EU-Project PROMOTE: A Process-oriented Approach for Knowledge Management. In: *PAKM 2000, Proceedings of the 3rd International Conference of Practical Aspects of Knowledge Management*, 2000, pp. 9–18

[KELLER 1985] KELLER, Arthur 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*. New York, NY, USA : ACM, 1985 (PODS '85). – ISBN 0–89791–153–9, pp. 154–163

[KELLER ET AL. 1992] KELLER, G.; NÜTTGENS, M. ; SCHEER, A.-W.: *Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK)*. Universität des Saarlandes, Institut für Wirtschaftsinformatik, 1992 (Institut für Wirtschaftsinformatik Saarbrücken: Veröffentlichungen des Instituts für Wirtschaftsinformatik - Heft 89)

[KELLER AND TEUFEL 1997] KELLER, Gerhard; TEUFEL, Thomas: *SAP R/3 prozessorientiert anwenden: iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten*. Addison-Wesley-Longman, 1997 (Edition SAP). – ISBN 9783827312587

[KELLY ET AL. 1996] KELLY, Steven; LYYTINEN, Kalle ; ROSSI, Matti: MetaEdit+ A fully configurable multi-user and multi-tool CASE and CAME environment. In: CONSTANTOPOULOS, Panos (Eds.); MYLOPOULOS, John (Eds.) ; VASSILIOU, Yannis (Eds.): *Advanced Information Systems Engineering* Bd. 1080. Springer Berlin / Heidelberg, 1996. – ISBN 978–3–540–61292–6, pp. 1–21

[KENT 2002] KENT, Stuart: Model Driven Engineering. In: BUTLER, Michael J. (Eds.); PETRE, Luigia (Eds.) ; SERE, Kaisa (Eds.): *IFM '02: Proceedings of the Third International Conference on Integrated Formal Methods*. London, UK : Springer-Verlag, 2002 (IFM '02). – ISBN 3–540–43703–7, pp. 286–298

[KINDLER 2004] KINDLER, Ekkart: On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In: DESEL, Jörg (Eds.); PERNICI, Barbara (Eds.) ; WESKE, Mathias (Eds.): *International Conference on Business Process Management (BPM 2004)*, Springer-Verlag, 2004, pp. 82–97

[KNUBLAUCH 2004] KNUBLAUCH, Holger: Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protégé/OWL. In: FRANKEL, David S. (Eds.); KENDALL, Elisa F. (Eds.) ; MCGUINNESS, Deborah L. (Eds.): *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004

[KOSSAK ET AL. 2012] KOSSAK, Felix; ILLIBAUER, Christa ; GEIST, Verena: Event-Based Gateways: Open Questions and Inconsistencies. In: MENDLING, Jan (Eds.); WEIDLICH, Matthias (Eds.): *Business Process Model and Notation* Bd. 125. Springer Berlin Heidelberg, 2012, pp. 53–67

[KRAMER 2007] KRAMER, Jeff: Is Abstraction the Key to Computing? In: *Communications of the ACM* 50 (2007), No. 4, pp. 36–42. – ISSN 0001–0782

[KRAMER AND FINKELSTEIN 1991] KRAMER, Jeff; FINKELSTEIN, Anthony: A Configurable Framework for Method and Tool Integration. In: ENDRES, Albert (Eds.); WEBER, Herbert

(Eds.): *Proceedings of European Symposium on Software development Environments and CASE Technology*, Springer-Verlag, 1991, pp. 233–257

[KRAMER ET AL. 2013] KRAMER, Max E.; BURGER, Erik ; LANGHAMMER, Michael: View-centric Engineering with Synchronized Heterogeneous Models. In: *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. New York, NY, USA : ACM, 2013 (VAO '13), pp. 5:1–5:6

[KRUCHTEN 1995] KRUCHTEN, Philippe: Architectural Blueprints - The "4+1" View Model of Software Architecture. In: *IEEE Software* 12 (1995), No. 6, pp. 42–50

[KRUCHTEN 1996] KRUCHTEN, Philippe: Software Architecture - A Rational Metamodel. In: *Joint Proceedings of the second International Software Architecture Workshop (ISAW-2) and International Workshop on Multiple Perspectives in Software Development (Viewpoints '96) on SIGSOFT '96 Workshops*. New York, NY, USA : Association for Computing Machinery (ACM), 1996, pp. 5–7

[KRUCHTEN 2000] KRUCHTEN, Philippe: *The Rational Unified Process: An Introduction*. 2nd. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2000. – ISBN 0201707101

[KÜHN 2004] KÜHN, Harald: *Methodenintegration im Business Engineering*, University of Vienna, Department of Knowledge Engineering, PhD thesis, 2004

[KÜHN ET AL. 1999] KÜHN, Harald; JUNGINGER, Stefan; KARAGIANNIS, Dimitris ; PETERSEN, Carsten: Metamodellierung im Geschäftsprozessmanagement: Konzepte, Erfahrungen und Potentiale. In: DESEL, Jörg (Eds.); POHL, Klaus (Eds.) ; SCHÜRR, Andy (Eds.): *Proceedings Modellierung, March 1999, Karlsruhe*, Teubner, 1999. – ISBN 3–519–00274–4, pp. 75–90

[KUSEL ET AL. 2012] KUSEL, Aangelika; MITSCH, Stefan; RETSCHITZEGGER, Werner; SCHWINGER, Weiland; MAYR, Ralph ; SCHÖNBÖCK, Johannes: Ontology-Driven Generation of Multi-View Modeling Tools. In: *Proceedings of 11th IASTED International Conference on Software Engineering*, ACTA Press, 2012 (SE 2012), pp. 45–51

[LAKHOUA AND RAHMOUNI 2011] LAKHOUA, Mohamed N.; RAHMOUNI, M.: Investigation of the methods of enterprise modeling. In: *African Journal of Business Management* 5 (2011), No. 16, pp. 6845–6852

[LANKHORST 2009] LANKHORST, Marc: Enterprise Architecture at Work: Modelling, Communication and Analysis. In: DIETZ, Jan (Eds.); PROPER, Erik (Eds.) ; TRIBOLET, José

(Eds.): *The Enterprise Engineering Series*. 2nd. Springer-Verlag Berlin Heidelberg, 2009. – ISBN 9783642013096

[DE LARA ET AL. 2013] LARA, Juan de; GUERRA, Esther ; CUADRADO, Jesús S.: Model-driven engineering with domain-specific meta-modelling languages. In: *Software & Systems Modeling* 14 (2013), No. 1, pp. 1–31. – ISSN 1619–1366

[LARKIN AND SIMON 1987] LARKIN, Jill H.; SIMON, Herbert A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. In: *Cognitive Science* 11 (1987), No. 1, pp. 65–100

[LAUDER ET AL. 2012] LAUDER, Marius; ANJORIN, Anthony; VARRÓ, Gergely ; SCHÜRR, Andy: Bidirectional Model Transformation with Precedence Triple Graph Grammars. In: VALLECILLO, Antonio (Eds.); TOLVANEN, Juha-Pekka (Eds.); KINDLER, Ekkart (Eds.); STÖRRLE, Harald (Eds.) ; KOLOVOS, Dimitris (Eds.): *Proceedings of the 8th European Conference on Modelling Foundations and Applications*. Berlin, Heidelberg : Springer-Verlag, 2012 (ECMFA'12). – ISBN 978–3–642–31490–2, pp. 287–302

[LEBLEBICI ET AL. 2014] LEBLEBICI, Erhan; ANJORIN, Anthony ; SCHÜRR, Andy: Developing eMoflon with eMoflon. In: DI RUSCIO, Davide (Eds.); VARRO, Daniel (Eds.): *Theory and Practice of Model Transformations* Bd. 8568. Springer International Publishing, 2014, pp. 138–145

[LECHTENBÖRGER 2003] LECHTENBÖRGER, Jens: 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*. New York, NY, USA : Association for Computing Machinery (ACM), 2003 (PODS '03). – ISBN 1–58113–670–6, pp. 49–55

[LENZ AND OBERWEIS 2003] LENZ, Kirsten; OBERWEIS, Andreas: Interorganizational Business Process Management with XML Nets. In: H. EHRIG, G. Rozenberg H. W. W. Reisig R. W. Reisig (Eds.): *Petri Net Technology for Communication-Based Systems, Advances in Petri Nets* Bd. 2472. Springer-Verlag, 2003, pp. 243–263

[LICHKA ET AL. 2002] LICHKA, Christian; KARAGIANNIS, Dimitris ; KÜHN, Harald: ADOscore: IT-gestützte Balanced Scorecard. In: *WISU - das wirtschaftsstudium*. 2002, pp. 915–918

[LINDLAND ET AL. 1994] LINDLAND, Odd I.; SINDRE, Guttorm ; SOLVBERG, Arne: Understanding Quality in Conceptual Modeling. In: *Software, IEEE* 11 (1994), No. 2, pp. 42–49. – ISSN 0740–7459

[LINSTONE 1984] LINSTONE, Harold A.: *Multiple perspectives for decision making: bridging the gap between analysis and action*. New York : North-Holland, 1984. – ISBN 978–0130500489

[LINSTONE 1989] LINSTONE, Harold A.: Multiple Perspectives: Concept, Applications, and User Guidelines. In: *Systems Practice* 2 (1989), No. 3, pp. 307–331

[LINSTONE 2003] LINSTONE, Harold A.: The Multiple Perspective Concept. In: GLENN, Jerome C. (Eds.); GORDON, Theodore J. (Eds.): *Futures Research Methodology - V2.0*. 2003

[LIU ET AL. 2007] LIU, Na; HOSKING, John ; GRUNDY, John: MaramaTatau: Extending a Domain Specific Visual Language Meta Tool with a Declarative Constraint Mechanism. In: COX, Philip (Eds.); HOSKING, John (Eds.): *IEEE Symposium on Visual Languages and Human-Centric Computing*, 2007, pp. 95–103

[LOCHMANN AND HESSELLUND 2009] LOCHMANN, Henrik; HESSELLUND, Anders: An Integrated View on Modeling with Multiple Domain-Specific Languages. In: BREU, Ruth (Eds.): *Proceedings of the IASTED International Conference Software Engineering (SE 2009)*, ACTA Press, 2009, pp. 1–10

[LOPEZ-HERREJON AND EGYED 2011] LOPEZ-HERREJON, R.E.; EGYED, A.: C2MV2: Consistency and Composition for Managing Variability in Multi-view Systems. In: *15th European Conference on Software Maintenance and Reengineering (CSMR)*, 2011, pp. 347–350

[LUJÁN-MORA ET AL. 2006] LUJÁN-MORA, Sergio; TRUJILLO, Juan ; SONG, Il-Yeol: A UML profile for multidimensional modeling in data warehouses. In: *Data & Knowledge Engineering* 59 (2006), No. 3, pp. 725–769

[LYYTINEN 1987] LYYTINEN, Kalle: A Taxonomic Perspective of Information Systems Development: Theoretical Constructs and Recommendations. In: BOLAND, R. (Eds.); HIRSCHHEIM, R. (Eds.): *Critical Issues in Information Systems*. New York, NY, USA : John Wiley & Sons, Inc., 1987, pp. 3–41

[MAES AND POELS 2007] MAES, Ann; POELS, Geert: Evaluating quality of conceptual modelling scripts based on user perceptions. In: *Data & Knowledge Engineering* 63 (2007), No. 3, pp. 701 – 724

[MAIER 2004] MAIER, R.: *Knowledge Management Systems: Information and Communication Technologies for Knowledge Management*. 2nd. Springer, 2004. – ISBN 3–540–20547–0

[MARCH AND SMITH 1995] MARCH, Salvatore T.; SMITH, Gerald F.: Design and natural science research on information technology. In: *Decision Support Systems* 15 (1995), No. 4, pp. 251–266. – ISSN 0167–9236

[MAYER ET AL. 1995] MAYER, Richard J.; MENZEL, Christopher P.; PAINTER, Michael K.; DEWITTE, Paula S.; BLINN, Thomas ; PERAKATH, Benjamin: Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report / Wright-Patterson Air Force Base, Human Resources Directorate Logistics Research Division, Ohio. 1995. – Forschungsbericht

[MCUMBER AND CHENG 2001] MCUMBER, William E.; CHENG, Betty H. C.: A General Framework for Formalizing UML with Formal Languages. In: MÜLLER, Hausi A. (Eds.); HARROLD, Mary J. (Eds.) ; SCHÄFER, Wilhelm (Eds.): *Proceedings of the 23rd International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2001 (ICSE '01). – ISBN 0–7695–1050–7, pp. 433–442

[MENDLING AND AALST 2007] MENDLING, Jan; AALST, Wil Van D.: Formalization and Verification of EPCs with OR-Joins Based on State and Context. In: KROGSTIE, John (Eds.); OPDAHL, Andreas L. (Eds.) ; SINDRE, Guttorm (Eds.): *Proceedings of the 19th Internation Conference on Advanced Information Systems Engineering (CAiSE 2007)*, Springer-Verlag, 2007, pp. 439–453

[MENS ET AL. 2005] MENS, Tom; STRAETEN, Ragnhild Van D. ; SIMMONDS, Jocelyn: A Framework for Managing Consistency of Evolving UML Models. In: YANG, Hongji (Eds.): *Software Evolution with UML and XML*, 2005

[MENZEL AND MAYER 2006] MENZEL, Christopher; MAYER, Richard J.: The IDEF Family of Langauges. In: BERNUS, Peter (Eds.); MERTINS, Kai (Eds.) ; SCHMIDT, Günter (Eds.): *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, 2006 (International Handbooks on Information Systems). – ISBN 978–3–540–25472–0, pp. 215–249

[MESEGUER AND PREECE 1996] MESEGUER, Pedro; PREECE, Alun D.: Assessing the Role of Formal Specifications in Verification and Validation of Knowledge-Based Systems. In: *Procedings of the Third International Conference on Achieving Quality in Software*, Chapman & Hall, 1996, pp. 317–328

[MEYER 1985] MEYER, Bertrand: On Formalism in Specifications. In: *IEEE Software* 2 (1985), No. 1, pp. 6–26

[MILLER AND MUKERJI 2003] MILLER, Joaquin; MUKERJI, Jishnu: *MDA Guide Version*

1.0.1. 2003. – http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf, last checked: 2013-07-12

[MIOTTO AND VARDANEGA 2009] MIOTTO, Eric; VARDANEGA, Tullio: On the integration of domain-specific and scientific bodies of knowledge in model driven engineering. In: *Workshop on the Definition, evaluation, and exploitation of modelling and computing standards for Real-Time Embedded Systems, STANDRTS'09*. Dublin, Ireland, 2009

[MOODY 2005] MOODY, Daniel L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. In: *Data & Knowledge Engineering* 55 (2005), No. 3, pp. 243–276. – ISSN 0169–023X

[MOODY 2009] MOODY, Daniel L.: The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In: *Software Engineering, IEEE Transactions on* 35 (2009), No. 6. – ISSN 0098–5589

[MUNKER ET AL. 2014] MUNKER, Florian; ALBERS, Albert; WAGNER, Daniel ; BEHRENDT, Matthias: Multi-View Modeling in SysML: Thematic Structuring for Multiple Thematic Views. In: AZAD, M. M. (Eds.); BARRY, W. B. (Eds.): *Proceedings of the Conference on Systems Engineering Research, CSER 2014, Redondo Beach, CA, USA* Bd. 28, Elsevier, 2014, pp. 531–538

[MYLOPOULOS 1982] MYLOPOULOS, John: Conceptual Modelling and Telos. In: LOUCOPOULOS, Zicari R. Pericles (Eds.): *Conceptual Modeling, Databases and Case: An Integrated View of Information Systems Development*, John Wiley & Sons, 1982, pp. 49–68

[NUNAMAKER ET AL. 1990] NUNAMAKER, Jay F. J.; CHEN, Minder ; PURDIN, Titus D. M.: Systems Development in Information Systems Research. In: *Journal of Management Information Systems* 7 (1990), No. 3, pp. 89–106. – ISSN 0742–1222

[NUSEIBEH ET AL. 2001] NUSEIBEH, Bashar; EASTERBROOK, Steve ; RUSSO, Alessandra: Making inconsistency respectable in software development. In: *Journal of Systems and Software* 58 (2001), No. 2, pp. 171–180

[NUSEIBEH ET AL. 1994a] NUSEIBEH, Bashar; FINKELSTEIN, Anthony; KRAMER, Jeff ; EASTERBROOK, Steve: Concurrent Software Engineering: Coordinating Distributed Viewpoints for Managing Inconsistency. In: *IEEE Colloquium on Issues of Co-Operative Working in Concurrent Engineering*, 1994, pp. 10/1–10/2

[NUSEIBEH ET AL. 1994b] NUSEIBEH, Bashar; KRAMER, Jeff ; FINKELSTEIN, Anthony: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. In: *IEEE Transactions on Software Engineering* 20 (1994), No. 10, pp. 760–773

[NÜTTGENS AND RUMP 2002] NÜTTGENS, Markus; RUMP, Frank J.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: DESEL, Jörg (Eds.); WESKE, Mathias (Eds.): *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen - Promise 2002*, Gesellschaft für Informatik, Bonn, 2002, pp. 64–77

[OBJECT MANAGEMENT GROUP (OMG) 2003] OBJECT MANAGEMENT GROUP (OMG): *OMG Model Driven Architecture (MDA), Version 1.0.1, Release Date: 2003-06-12*. 2003. – http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf, last checked: 2013-08-07

[OBJECT MANAGEMENT GROUP (OMG) 2008] OBJECT MANAGEMENT GROUP (OMG): *OMG Business Process Model and Notation (BPMN) Specification, Version 1.1, Release Date: 2008-01-17*. 2008. – http://www.omg.org/spec/BPMN/1.1/PDF, last checked: 2013-08-07

[OBJECT MANAGEMENT GROUP (OMG) 2010] OBJECT MANAGEMENT GROUP (OMG): *OMG Object Constraint Language (OCL) Specification, Version 2.2, Release Date: 2010-02-01*. 2010. – http://www.omg.org/spec/OCL/2.2, last checked: 2013-07-28

[OBJECT MANAGEMENT GROUP (OMG) 2011a] OBJECT MANAGEMENT GROUP (OMG): *OMG Business Process Model and Notation (BPMN) Specification, Version 2.0, Release Date: 2011-01-03*. 2011. – http://www.omg.org/spec/BPMN/2.0/PDF, last checked: 2013-08-01

[OBJECT MANAGEMENT GROUP (OMG) 2011b] OBJECT MANAGEMENT GROUP (OMG): *OMG MetaObject Facility (MOF) Core Specification, Version 2.4.1, Release Date: 2013-06-01*. 2011. – http://www.omg.org/spec/MOF/2.4.1/PDF/, last checked: 2013-07-28

[OBJECT MANAGEMENT GROUP (OMG) 2011c] OBJECT MANAGEMENT GROUP (OMG): *OMG Unified Modeling Language (UML) Infrastructure Specification, Version 2.4.1, Release Date: 2011-08-05*. 2011. – http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/, last checked: 2013-11-02

[OBJECT MANAGEMENT GROUP (OMG) 2011d] OBJECT MANAGEMENT GROUP (OMG): *OMG Unified Modeling Language (UML) Superstructure Specification, Version 2.4.1, Release Date: 2011-08-06*. 2011. – http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF, last checked: 2013-10-27

[OBJECT MANAGEMENT GROUP (OMG) 2012a] OBJECT MANAGEMENT GROUP (OMG):

OMG Diagram Definition (DD) Specification Version 1.0. 2012. – http://www.omg.org/spec/DD/1.0/, last checked: 2013-10-27

[OBJECT MANAGEMENT GROUP (OMG) 2012b] OBJECT MANAGEMENT GROUP (OMG): *OMG Systems Modeling Language (SysML) Specification, Version 1.3, Release Date: 2012-06-01*. 2012. – http://www.omg.org/spec/SysML/1.3/, last checked: 2013-10-23

[OBJECT MANAGEMENT GROUP (OMG) 2013] OBJECT MANAGEMENT GROUP (OMG): *Semantics Of A Foundational Subset For Executable UML Models (FUML) Specification Version 1.1, Release Date: 2013-08-06*. 2013. – http://www.omg.org/spec/FUML/1.1, last checked: 2013-11-03

[OBJECT MANAGEMENT GROUP (OMG) 2015] OBJECT MANAGEMENT GROUP (OMG): *OMG Query/View/Transformation (QVT) Specification, Version 1.2, Release Date: 2015-02-01*. 2015. – http://www.omg.org/spec/QVT/1.2/PDF, last checked: 2015-04-10

[OBRST 2003] OBRST, Leo: Ontologies for Semantically Interoperable Systems. In: *Proceedings of the twelfth International Conference on Information and Knowledge Management*. New York, NY, USA : Association for Computing Machinery (ACM), 2003 (CIKM '03). – ISBN 1–58113–723–0, pp. 366–369

[OLIGA 1988] OLIGA, John C.: Methodological Foundations of Systems Methodologies. In: *Systems Practice* 1 (1988), No. 1, pp. 87–112

[ÖSTERLE ET AL. 2010] ÖSTERLE, Hubert; BECKER, Jörg; FRANK, Ulrich; HESS, Thomas; KARAGIANNIS, Dimitris; KRCMAR, Helmut; LOOS, Peter; MERTENS, Peter; OBERWEIS, Andreas ; SINZ, Elmar J.: Memorandum on design-oriented information systems research. In: *European Journal of Information Systems* 20 (2010), No. 1, pp. 7–10

[OUYANG ET AL. 2006] OUYANG, Chun; DUMAS, Marlon; HOFSTEDE, Arthur H. ; AALST, Wil M. d.: From BPMN process models to BPEL web services. In: *ICWS'06. International Conference on Web Services* IEEE, 2006, pp. 285–292

[PEFFERS ET AL. 2012] PEFFERS, Ken; ROTHENBERGER, Marcus; TUUNANEN, Tuure ; VAEZI, Reza: Design Science Research Evaluation. In: PEFFERS, Ken (Eds.); ROTHENBERGER, Marcus (Eds.) ; KUECHLER, Bill (Eds.): *Design Science Research in Information Systems. Advances in Theory and Practice* Bd. 7286. Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–29862–2, pp. 398–410

[PEFFERS ET AL. 2007] PEFFERS, Ken; TUUNANEN, Tuure; ROTHENBERGER, Marcus ; CHATTERJEE, Samir: A Design Science Research Methodology for Information Systems Research. In: *Journal of Management Information Systems* 24 (2007), No. 3, pp. 45–77. – ISSN 0742–1222

[PERSSON ET AL. 2013] PERSSON, Magnus; TÖRNGREN, Martin; QAMAR, Ahsan; WESTMAN, Jonas; BIEHL, Matthias; TRIPAKIS, Stavros; VANGHELUWE, Hans ; DENIL, Joachim: 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)*, 2013, pp. 1–10

[PETRI 1962] PETRI, Carl A.: *Kommunikation mit Automaten*. Technische Universität Darmstadt, Institut für Instrumentelle Mathematik, PhD thesis, 1962

[PETRI 1966] PETRI, Carl A.: *Communication with automata*, Technische Universität Darmstadt, PhD thesis, 1966. – approved english translation by Clifford F. Greene

[PFEIFFER AND WASOWSKI 2012a] PFEIFFER, Rolf-Helge; WASOWSKI, Andrzej: Cross-Language Support Mechanisms Significantly Aid Software Development. In: FRANCE, Robert B. (Eds.); KAZMEIER, Jürgen (Eds.); BREU, Ruth (Eds.) ; ATKINSON, Colin (Eds.): *Model Driven Engineering Languages and Systems* Bd. 7590. Springer Berlin Heidelberg, 2012, pp. 168–184

[PFEIFFER AND WASOWSKI 2012b] PFEIFFER, Rolf-Helge; WASOWSKI, Andrzej: TexMo: A Multi-language Development Environment. In: VALLECILLO, Antonio (Eds.); TOLVANEN, Juha-Pekka (Eds.); KINDLER, Ekkart (Eds.); STÖRRLE, Harald (Eds.) ; KOLOVOS, Dimitris (Eds.): *Modelling Foundations and Applications* Bd. 7349. Springer Berlin Heidelberg, 2012, pp. 178–193

[PFEIFFER AND WASOWSKI 2015] PFEIFFER, Rolf-Helge; WASOWSKI, Andrzej: The design space of multi-language development environments. In: *Software & Systems Modeling* 14 (2015), No. 1, pp. 383–411. – ISSN 1619–1366

[PRIES-HEJE ET AL. 2008] PRIES-HEJE, Jan R.; BASKERVILLE, Richard ; VENABLE, John C.: Strategies for design science research evaluation. In: ENGELS, Gregor (Eds.); KARAGIANNIS, Dimitris (Eds.) ; MAYR, Heinrich C. (Eds.): *Proceedings 16th European Conference on Information Systems (ECIS)*. Galway, Ireland, 2008, pp. 1–12

[PÜTZ AND SINZ 2010a] PÜTZ, Corinna; SINZ, Elmar J.: Model-driven Derivation of BPMN Workflow Schemata from SOM Business Process Models. In: *Enterprise Modelling and Information Systems Architectures (EMISA)* 5 (2010), No. 2, pp. 57–72

[PÜTZ AND SINZ 2010b] PÜTZ, Corinna; SINZ, Elmar J.: Modellgetriebene Ableitung von BPMN-Workflowschemata aus SOM-Geschäftsprozessmodellen. In: ENGELS, Gregor (Eds.); KARAGIANNIS, Dimitris (Eds.) ; MAYR, Heinrich C. (Eds.): *Proceedings Modellierung 2010. March 24-26, 2010, Klagenfurt, Austria* Bd. 161. Bonn : Gesellschaft für Informatik - GI, 2010 (GI-Edition Lecture Notes in Informatics, Proceedings). – ISBN 978–3–88579–255–0, pp. 253–268

[QAMAR ET AL. 2013] QAMAR, Ahsan; HERZIG, Sebastian ; PAREDIS, Christiaan J. J.: A Domain-Specific Language for Dependency Management in Model-Based Systems Engineering. In: JACQUET, Christophe (Eds.); BALASUBRAMANIAN, Daniel (Eds.); JONES, Edward (Eds.) ; MÉSZÁROS, Tamás (Eds.): *Proceedings of the 7th Workshop on Multi-Paradigm Modeling (MPM 2013)*. Aachen, 2013. – ISSN 1613–0073, pp. 7–16

[QURESHI 2012] QURESHI, Tahir N.: *Enhancing Model-Based Development of Embedded Systems*, KTH Royal Institute of Technology, Stockholm, PhD thesis, 2012

[RADJENOVIC AND PAIGE 2008] RADJENOVIC, Alek; PAIGE, Richard F.: The Role of Dependency Links in Ensuring Architectural View Consistency. In: *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. Washington, DC, USA : IEEE Computer Society, 2008 (WICSA '08). – ISBN 978–0–7695–3092–5, pp. 199–208

[RAYMOND 1995] RAYMOND, Kerry: Reference Model of Open Distributed Processing (RM-ODP): Introduction. In: RAYMOND, Kerry (Eds.); ARMSTRONG, Liz (Eds.): *Open Distributed Processing*. Springer US, 1995 (IFIP - The International Federation for Information Processing). – ISBN 978–1–4757–6074–3, pp. 3–14

[REDDY ET AL. 1994] REDDY, MP; PRASAD, Bandreddi E.; REDDY, PG ; GUPTA, A: A methodology for integration of heterogeneous databases. In: *Knowledge and Data Engineering, IEEE Transactions on* 6 (1994), No. 6, pp. 920–933

[REEVES ET AL. 1995] REEVES, Andrew; MARASHI, Mustafa ; BUDGEN, David: A software design framework or how to support real designers. In: *Software Engineering Journal* 10 (1995), No. 4, pp. 141–155. – ISSN 0268–6961

[REINEKE AND TRIPAKIS 2014] REINEKE, Jan; TRIPAKIS, Stavros: Basic Problems in Multi-View Modeling / EECS Department, University of California, Berkeley. 2014 (UCB/EECS-2014-4). – Forschungsbericht

[REITER 1991] REITER, Raymond: The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In: LIFSCHITZ,

Vladimir (Eds.): *Artificial Intelligence and Mathematical Theory of Computation*. San Diego, CA, USA : Academic Press Professional, Inc., 1991. – ISBN 0–12–450010–2, pp. 359–380

[RICHTERS AND GOGOLLA 1998] RICHTERS, Mark; GOGOLLA, Martin: On Formalizing the UML Object Constraint Language OCL. In: LING, Tok W. (Eds.); RAM, Sudha (Eds.) ; LEE, Mong-Li (Eds.): *Procedings of the 17th International Conference on Conceptual Modeling*, Springer, 1998 (ER'98), pp. 449–464

[ROMEIKAT ET AL. 2008] ROMEIKAT, Raphael; ROSER, Stephan; MÜLLENDER, Pascal ; BAUER, Bernhard: Translation of QVT Relations into QVT Operational Mappings. In: VALLECILLO, Antonio (Eds.); GRAY, Jeff (Eds.) ; PIERANTONIO, Alfonso (Eds.): *Theory and Practice of Model Transformations* Bd. 5063. Springer Berlin Heidelberg, 2008, pp. 137–151

[ROMERO ET AL. 2009] ROMERO, José R.; JAEN, Juan I. ; VALLECILLO, Antonio: Realizing Correspondences in Multi-Viewpoint Specifications. In: GUERRERO, Juan E. (Eds.): *Proceedings of the 13th International Enterprise Distributed Object Computing Conference, EDOC '09*, 2009, pp. 163–172

[ROSS AND SCHOMAN 1977] ROSS, Douglas T.; SCHOMAN, Kenneth E. J.: Structured Analysis for Requirements Definition. In: *IEEE Transactions on Software Engineering* SE-3 (1977), No. 1, pp. 6–15

[ROZENBERG 1997] ROZENBERG, Grzegorz: *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. River Edge, NJ, USA : World Scientific Publishing Company, 1997. – ISBN 98–102288–48

[RUMBAUGH ET AL. 1990] RUMBAUGH, James R.; BLAHA, Michael R.; LORENSEN, William; EDDY, Frederick ; PREMERLANI, William: *Object-Oriented Modeling and Design*. 1st. Prentice-Hall, 1990. – ISBN 0136298419

[SANDERS 1999] SANDERS, William H.: Integrated frameworks for multi-level and multi-formalism modeling. In: *The 8th International Workshop on Petri Nets and Performance Models, 1999. Proceedings*, 1999. – ISSN 1063–6714, pp. 2–9

[SCHAMAI ET AL. 2009] SCHAMAI, Wladimir; FRITZSON, Peter; PAREDIS, Chris ; POP, Adrian: Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. In: CASELLA, Francesco (Eds.): *Proceedings of the 7th Modelica Conference*, Linköping University Electronic Press, 2009, pp. 612–621

[SCHEER 1992] SCHEER, August W.: *Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung*. Springer, 1992. – ISBN 978–3540554011

[SCHEER 2000] SCHEER, August-Wilhelm: *Aris-Business Process Modeling*. 3rd. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2000. – ISBN 3540658351

[SCHEER AND SCHNEIDER 2006] SCHEER, August-Wilhelm; SCHNEIDER, Kristof: ARIS - Architecture of Integrated Information Systems. In: BERNUS, Peter (Eds.); MERTINS, Kai (Eds.) ; SCHMIDT, Günter (Eds.): *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, 2006 (International Handbooks on Information Systems). – ISBN 978–3–540–25472–0, pp. 605–623

[SCHMIDT 2006] SCHMIDT, Douglas C.: Guest Editor's Introduction: Model-Driven Engineering. In: *Computer* 39 (2006), Februar, No. 2, pp. 25–31. – ISSN 0018–9162

[SCHÖNTHALER ET AL. 2012] SCHÖNTHALER, Frank; VOSSEN, Gottfried ; OBERWEIS, Andreas: *Business Processes for Business Communities: Modeling Languages, Methods, Tools*. Springer, 2012. – ISBN 978–3642–24790–3

[SCHWAB ET AL. 2010] SCHWAB, Margit; KARAGIANNIS, Dimitris ; BERGMAYR, Alexander: i* on ADOxx®: A Case Study. In: *iStar'10, Proceedings of the 4th International i* Workshop*, 2010, pp. 92–97

[SEIDEWITZ 2013] SEIDEWITZ, Ed: *UML: Once More with Meaning*. 2013. – Invited talk at the University of Maryland, 2013/04/15, `http://www.isr.umd.edu/sites/default/files/Seidewitz\_041513.pptx`, last checked: 2013-10-27

[SEIN ET AL. 2011] SEIN, Maung K.; HENFRIDSSON, Ola; PURAO, Sandeep; ROSSI, Matti ; LINDGREN, Rikard: Action Design Research. In: *MIS Quaterly* 35 (2011), März, No. 1, pp. 37–56. – ISSN 0276–7783

[SELIC 2007] SELIC, Bran: A Systematic Approach to Domain-Specific Language Design Using UML. In: *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*. Washington, DC, USA : IEEE Computer Society, 2007 (ISORC '07). – ISBN 0–7695–2765–5, pp. 2–9

[SHAH ET AL. 2010] SHAH, Aditya A.; KERZHNER, Aleksandr A.; SCHAEFER, Dirk ; PAREDIS, Christiaan J.: Multi-view Modeling to Support Embedded Systems Engineering in SysML. In: ENGELS, Gregor (Eds.); LEWERENTZ, Claus (Eds.); SCHÄFER, Wilhelm (Eds.); SCHÜRR, Andy (Eds.) ; WESTFECHTEL, Bernhard (Eds.): *Graph Transformations and Model-Driven Engineering* Bd. 5765. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–17321–9, pp. 580–601

[SHEN ET AL. 2004] SHEN, Hui; WALL, Brian; ZAREMBA, Michal; CHEN, Yuliu ; BROWNE, Jim: Integration of business modelling methods for enterprise information system analysis and user requirements gathering. In: *Computers in Industry* 54 (2004), No. 3, pp. 307–323

[SHETH AND LARSON 1990] SHETH, Amit P.; LARSON, James A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In: *ACM Computing Surveys* 22 (1990), No. 3, pp. 183–236. – ISSN 0360–0300

[SIMON 1996] SIMON, Herbert A.: *The Sciences of the Artificial*. 3. MIT press, 1996. – ISBN 0–262–19374–4

[SINDEREN 1995] SINDEREN, Marten J.: *On the design of application protocols*, University of Twente, PhD thesis, 1995

[SINZ 1987] SINZ, Elmar J.: *Datenmodellierung betrieblicher Probleme und ihre Unterstützung durch ein wissensbasiertes Entwicklungssystem, habilitation thesis*. 1987

[SINZ 1988] SINZ, Elmar J.: Das Strukturierte Entity-Relationship-Modell (SER-Modell). In: *Angewandte Informatik* 30 (1988), No. 5, pp. 191–202

[SINZ 1996] SINZ, Elmar J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme. Entwicklung, aktueller Stand und Trends. In: HEILMANN, Heidi (Eds.); HEINRICH, Lutz J. (Eds.) ; ROITHMAYR, Friedrich (Eds.): *Information Engineering. Wirtschaftsinformatik im Schnittpunkt von Wirtschafts-, Sozial- und Ingenieurwissenschaften*. Oldenbourg Verlag, München, Wien, 1996, pp. 123–143

[SINZ 1997] SINZ, Elmar J.: Architektur betrieblicher Informationssysteme. In: *Bamberger Beiträge zur Wirtschaftsinformatik* (1997), No. 40

[SINZ 2002] SINZ, Elmar J.: Architektur von Informationssystemen. In: RECHENBERG, Peter (Eds.); POMBERGER, Gustav (Eds.): *Informatik-Handbuch, 3rd updated and extended edition*. Hanser, München, Wien, 2002, pp. 1055–1068

[SMITH AND WELTY 2001] SMITH, Barry; WELTY, Christopher: Ontology: Towards a New Synthesis. In: *Proceedings of the International Conference on Formal Ontology in Information Systems*. New York, NY, USA : Association for Computing Machinery (ACM), 2001 (FOIS '01). – ISBN 1–58113–377–4

[SONI ET AL. 1996] SONI, Dilip; NORD, Robert L.; HSU, Liang ; DRONGOWSKI, Paul J.: Many Faces of Software Architecture. In: LAMB, David A. (Eds.): *Studies of Software Design* Bd. 1078. Springer Berlin Heidelberg, 1996, pp. 6–16

[STACHOWIAK 1973] STACHOWIAK, Herbert: *Allgemeine Modelltheorie*. Springer-Verlag, 1973. – ISBN 9783211811061

[STARK AND ESSWEIN 2012] STARK, Jeannette; ESSWEIN, Werner: Rules from Cognition for Conceptual Modelling. In: ATZENI, Paolo (Eds.); CHEUNG, David (Eds.) ; RAM, Sudha (Eds.): *Proceedings of the 2012 International Conference on Conceptual Modeling, ER 2012* Bd. 7532. Springer Berlin Heidelberg, 2012, pp. 78–87

[STRECKER ET AL. 2012] STRECKER, Stefan; FRANK, Ulrich; HEISE, David ; KATTENSTROTH, Heiko: METRICM: a modeling method in support of the reflective design and use of performance measurement systems. In: *Information Systems and e-Business Management* 10 (2012), No. 2, pp. 241–276. – ISSN 1617–9846

[STRECKER ET AL. 2011] STRECKER, Stefan; HEISE, David ; FRANK, Ulrich: RISKM: A multi-perspective modeling method for IT risk assessment. In: *Information Systems Frontiers* 13 (2011), No. 4, pp. 595–611. – ISSN 1387–3326

[SUSMAN AND EVERED 1978] SUSMAN, Gerald I.; EVERED, Roger D.: An Assessment of the Scientific Merits of Action Research. In: *Administrative Science Quarterly* 23 (1978), No. 4, pp. 582–603. – ISSN 00018392

[SUTCLIFFE 2002] SUTCLIFFE, Alistair: *Domain Theory: Patterns for Knowledge and Software Reuse*. Hillsdale, NJ, USA : L. Erlbaum Associates Inc., 2002. – ISBN 0805839518

[SZEGHEO AND ANDERSEN 1999] SZEGHEO, Orsolya; ANDERSEN, Bjørn: Modeling the Extended Enterprise: A Comparison of Different Modeling Approaches. In: *Proceedings of International Conference on Enterprise Modelling (IEMC'99), Verdal, Norge, June 14-16* Productivity Press, 1999

[TEEUW AND VAN DEN BERG 1997] TEEUW, WB; BERG, H Van d.: On the quality of conceptual models. In: *Proceedings of the ER'97 Workshop on Behavioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling* Bd. 97. UCLA, Los Angeles, California, 1997, pp. 6–7

[TEUSCH AND SINZ 2012] TEUSCH, Andree; SINZ, Elmar J.: Konzeptuelle Modellierung partieller SOA. In: MATTFELD, Dirk C. (Eds.); ROBRA-BISSANTZ, Susanne (Eds.): *Proceedings Multikonferenz Wirtschaftsinformatik 2012 Braunschweig*, 2012 (MKWI 2012), pp. 1637 – 1648

[THE OPEN GROUP 2011] THE OPEN GROUP: *The Open Group Architecture Framework (TOGAF) Specification, Version 9.1, Release Date: 2011-07-21*. 2011. – `https://www.opengroup.org/togaf/`, last checked: 2015-03-31

[THE OPEN GROUP 2012] THE OPEN GROUP: *The Open Group ArchiMate Specification, Version 2.1, Release Year: 2012*. 2012. – http://www.opengroup.org/subjectareas/enterprise/archimate, last checked: 2015-03-31

[THOMAS AND FELLMANN 2007] THOMAS, Oliver; FELLMANN, Michael: Semantic EPC: Enhancing Process Modeling Using Ontology Languages. In: HEPP, M. (Eds.); HINKELMANN, K. (Eds.); KARAGIANNIS, D. (Eds.); KLEIN, R. (Eds.) ; STOJANOVIC, N. (Eds.): *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, RWTH Aachen, 2007

[TOLVANEN AND ROSSI 2003] TOLVANEN, Juha-Pekka; ROSSI, Matti: 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*. New York, NY, USA : Association for Computing Machinery (ACM), 2003 (OOPSLA '03), pp. 92–93

[TSALGATIDOU AND LOUCOPOULOS 1991] TSALGATIDOU, A.; LOUCOPOULOS, P.: Rule-based behaviour modelling: specification and validation of information systems dynamics. In: *Information and Software Technology* 33 (1991), No. 6, pp. 425–432. – ISSN 0950–5849

[TSICHRITZIS 1997] TSICHRITZIS, Dennis: Beyond Calculation: The Next Fifty Years of Computing. In: DENNING, Peter J. (Eds.); METCALFE, Robert M. (Eds.): *The Dynamics of Innovation*. New York, NY, USA : Copernicus, 1997. – ISBN 0–38794932–1, pp. 259–265

[UNION 1997] UNION, International T.: *Information technology - Open distributed processing - Reference Model: Overview*. 1997. – online https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.901-199708-I!!PDF-E&type=items, last checked: 2014-05-08

[USMAN ET AL. 2008] USMAN, Muhammad; NADEEM, Aamer; KIM, Tai-hoon ; CHO, Eun-suk: A Survey of Consistency Checking Techniques for UML Models. In: *Proceedings of the 2008 Advanced Software Engineering and Its Applications*. Washington, DC, USA : IEEE Computer Society, 2008 (ASEA '08). – ISBN 978–0–7695–3432–9, pp. 57–62

[VAN AKEN 2005] VAN AKEN, Joan E.: Management research as a design science: Articulating the research products of mode 2 knowledge production in management. In: *British journal of management* 16 (2005), No. 1, pp. 19–36

[VARRÓ AND BALOGH 2007] VARRÓ, Dániel; BALOGH, András: The Model Transformation Language of the VIATRA2 Framework. In: *Scientific Computer Programming* 68 (2007), No. 3, pp. 187–207

[VENABLE ET AL. 2012] VENABLE, John R.; PRIES-HEJE, Jan ; BASKERVILLE, Richard: A Comprehensive Framework for Evaluation in Design Science Research. In: PEFFERS, Ken (Eds.); ROTHENBERGER, Marcus A. (Eds.) ; JR., William L. K. (Eds.): *Proceedings of the 7th International Conference on Design Science Research in Information Systems, Advances in Theory and Practice*, Springer, 2012 (DESRIST), pp. 423–438

[VESSEY 1991] VESSEY, Iris: Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature. In: *Decision Sciences* 22 (1991), No. 2, pp. 219–240. – ISSN 1540–5915

[VITTORINI ET AL. 2004] VITTORINI, Valeria; IACONO, Mauro; MAZZOCCA, Nicola ; FRANCESCHINIS, Giuliana: The OsMoSys approach to multi-formalism modeling of systems. In: *Software and Systems Modeling* 3 (2004), No. 1, pp. 68–81. – ISSN 1619–1366

[VOELTER ET AL. 2013] VOELTER, Markus; RATIU, Daniel ; TOMASSETTI, Federico: Requirements as First-Class Citizens: Integrating Requirements closely with Implementation Artifacts. In: OBER, Iulian (Eds.); NOYRIT, Florian (Eds.); GRAF, Susanne (Eds.) ; KARSAI, Gabor (Eds.): *Proceedings of the 6th International Workshop on Model Based Architecting and Construction of Embedded Systems (ACESMB 2013)*. Miami, Florida, 2013. – ISBN 978–0–470–02570–3

[VÖLTER ET AL. 2013] VÖLTER, Markus; STAHL, Thomas; BETTIN, Jorn; HAASE, Arno ; HELSEN, Simon: *Model-driven software development: technology, engineering, management*. John Wiley & Sons, 2013

[VOSSEN 1994] VOSSEN, Gottfried: *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme*. 2nd edition. Addison-Wesley, Bonn, 1994. – ISBN 978–3893195664

[WAGNER AND FERSTL 2010] WAGNER, Daniel; FERSTL, Otto K.: Erhöhte Abbildungstreue von Geschäftsprozessmodellen durch Kontextsensitivität. In: ENGELS, Gregor (Eds.); KARAGIANNIS, Dimitris (Eds.) ; MAYR, Heinrich C. (Eds.): *Proceedings Modellierung 2010. March 24-26, 2010, Klagenfurt, Austria* Bd. 161, Gesellschaft für Informatik - GI, 2010, pp. 117–132

[WALLS ET AL. 1992] WALLS, Joseph G.; WIDMEYER, George R. ; EL SAWY, Omar A.: Building an Information System Design Theory for Vigilant EIS. In: *Information Systems Research* 3 (1992), No. 1, pp. 36–59

[WALTER AND EBERT 2009] WALTER, Tobias; EBERT, Jürgen: Combining DSLs and Ontologies Using Metamodel Integration. In: TAHA, Walid M. (Eds.): *Domain-Specific Languages* Bd. 5658. Springer Berlin Heidelberg, 2009. – ISBN 978–3–642–03033–8, pp. 148–169

[WALTER ET AL. 2009] WALTER, Tobias; SILVA PARREIRAS, Fernando ; STAAB, Steffen: OntoDSL: An Ontology-Based Framework for Domain-Specific Languages. In: SCHÜRR, Andy (Eds.); SELIC, Bran (Eds.): *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*. Berlin, Heidelberg : Springer-Verlag, 2009 (MODELS '09). – ISBN 978–3–642–04424–3, pp. 408–422

[WAND AND WEBER 2002] WAND, Y.; WEBER, R.: Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. In: *Information Systems Research* 13 (2002), No. 4, pp. 363–376

[WANG ET AL. 1997] WANG, Enoch Y.; RICHTER, Heather A. ; CHENG, Betty H. C.: Formalizing and Integrating the Dynamic Model within OMT. In: ADRION, W. R. (Eds.); FUGGETTA, Alfonso (Eds.); TAYLOR, Richard N. (Eds.) ; WASSERMAN, Anthony I. (Eds.): *Proceedings of the 19th International Conference on Software Engineering*. New York, NY, USA : ACM, 1997 (ICSE '97). – ISBN 0–89791–914–9, pp. 45–55

[WEBER 1997] WEBER, Ron: *Ontological foundations of information systems*. Coopers & Lybrand and the Accounting Association of Australia and New Zealand Melbourne, 1997. – ISBN 1321–2605

[WEIDLICH ET AL. 2010] WEIDLICH, Matthias; DIJKMAN, Remco ; MENDLING, Jan: The ICoP Framework: Identification of Correspondences between Process Models. In: PERNICI, Barbara (Eds.): *Advanced Information Systems Engineering* Bd. 6051. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–13093–9, pp. 483–498

[WHITE, STEPHEN A. 2005] WHITE, STEPHEN A.: *Using BPMN to Model a BPEL Process*. 2005. – `http://www.omg.org/bpmn/Documents/Mapping\_BPMN\_to\_BPEL\_Example.pdf`, last checked: 2013-08-07

[WHITTEN ET AL. 2004] WHITTEN, Jeffrey L.; BARLOW, Victor M. ; BENTLEY, Lonnie: *Systems analysis and design methods*. McGraw-Hill Professional, Irwin, Boston, 2004. – ISBN 978–0073052335. – 6th edition

[WIERINGA 2012] WIERINGA, Roel: Designing Technical Action Research and Generalizing from Real-World Cases. In: RALYTÉ, Jolita (Eds.); FRANCH, Xavier (Eds.); BRINKKEMPER, Sjaak (Eds.) ; WRYCZA, Stanislaw (Eds.): *Proceedings of the 24th Internation Conference on Computer Aided Software Engineering - Advanced Information Systems Engineering* Bd. 7328. Springer Berlin Heidelberg, 2012, pp. 697–698

[WIERINGA AND MORALI 2012] WIERINGA, Roel; MORALI, Ayşe: Technical Action Research as a Validation Method in Information Systems Design Science. In: PEFFERS, Ken

(Eds.); ROTHENBERGER, Marcus (Eds.) ; KUECHLER, Bill (Eds.): *Proceedings of the 7th International Conference on Design Science Research in Information Systems. Advances in Theory and Practice* Bd. 7286. Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–29862–2, pp. 220–238

[WOLF AND BENKER 2013] WOLF, Matthias; BENKER, Thomas: Vom SOM-Geschäftsprozessmodell zur vollständig dokumentenorientierten RESTful SOA - Ein modellbasierter Ansatz. In: ALT, Rainer (Eds.); FRANCZYK, Bogdan (Eds.): *Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013)*, 2013, pp. 1229–1243

[WOOD ET AL. 2008] WOOD, S.K.; AKEHURST, D.H.; UZENKOV, O.; HOWELLS, W.G.J. ; MCDONALD-MAIER, K.D.: A Model-Driven Development Approach to Mapping UML State Diagrams to Synthesizable VHDL. In: *IEEE Transactions on Computers* 57 (2008), No. 10, pp. 1357–1371. – ISSN 0018–9340

[WOOD-HARPER 1982] WOOD-HARPER, A. T.: A summary of a methodology for the analysis and design of information systems for small organizations. In: BJORN-ANDERSEN, N. (Eds.): *Towards Tools for Transition Systems Design Methodologies*. 1982

[WOOD-HARPER ET AL. 1985] WOOD-HARPER, A. T.; ANTILL, Lyn ; AVISON, David E.: *Information Systems Definition: the Multiview Approach*. Oxford, UK, UK : Blackwell Scientific Publications, Ltd., 1985. – ISBN 0–632–01216–8

[WOOD-HARPER AND AVISON 1992] WOOD-HARPER, A. T.; AVISON, David E.: Reflections from the experience of using Multiview: through the lens of software systems methodology. In: *Systemist* 14 (1992), No. 3

[YIE ET AL. 2009] YIE, Andrés; CASALLAS, Rubby; DERIDDER, Dirk ; WAGELAAR, Dennis: A practical approach to multi-modeling views composition. In: *Proceedings of the 3rd International Workshop on Multi-Paradigm Modeling (MPM 2009)*. Denver, Colorado, USA : ECEASST, 2009 (Electronic Communications of the EASST)

[ZACHMAN 1987] ZACHMAN, John A.: A Framework for Information Systems Architecture. In: *IBM Systems Journal* 26 (1987), No. 3, pp. 276–292

[ZHAO ET AL. 2006] ZHAO, Dejin; GRUNDY, John ; HOSKING, John: Generating Mobile Device User Interfaces for Diagram-based Modelling Tools. In: *Proceedings of the 7th Australasian User Interface Conference - Volume 50*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2006 (AUIC '06), pp. 101–108

[ZHU ET AL. 2014] ZHU, Xiuna; MOU, Dongyue ; RATIU, Daniel: Structured Multi-view Modeling by Tabular Notation. In: GORSCHEK, Tony (Eds.); LUTZ, Robyn R. (Eds.): *IEEE*

22nd International Requirements Engineering Conference, RE Karlskrona, Sweden, IEEE, 2014, pp. 327–328