Formal Aspects of Enterprise Modeling Methods: A Comparison Framework

Domenik Bork
University of Vienna
Research Group Knowledge Engineering
domenik.bork@univie.ac.at

Hans-Georg Fill
University of Vienna
Research Group Knowledge Engineering
hans-georg.fill@univie.ac.at

Abstract

For the design of work and knowledge systems it is today common to revert to enterprise modeling methods. These methods not only support the representation and analysis of complex interactions between technical services and human actors. The resulting models also provide value through acting as knowledge bases themselves. Thereby, the formalization of modeling methods is essential to unambiguously define their structure, behavior, and semantics, and enable an intersubjective understanding and machine-processability.

In this paper we analyze and compare six common enterprise modeling methods in regard to the formalization of their process-related aspects. From this comparison we derive implications for choosing an appropriate method when designing work and knowledge systems.

1. Introduction

When designing advanced knowledge systems 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 [1]. These dimensions include environmental factors such as the globalization of businesses, increasing employee mobility or fierce international competition, as well as technical aspects such as rapid and frequent changes in information and communication technologies, c.f. [2]. 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 [3]. Conceptual modeling produces a "common understanding" and is used "as a communication, analysis, and documentation tool for domain knowledge and IS requirements" [4, p. 702]. Furthermore, it provides input for the system design process [4, 3]. In addition, the models provide value themselves by acting as machine-processable *knowledge bases* for answering queries, simulating behavior, performing reasoning, verification & validation, or generating executable code [5].

For realizing this additional *model value* it is important to provide sound and intersubjectively 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 interoperability 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" [6, p. 321]. Thus it becomes necessary to provide *formalized*, i.e. unambiguous, specifications of enterprise modeling methods. For some of these methods such specifications have been available already at the time of their introduction, whereas for others such formal specifications have been added later or are not yet available – see e.g. the recent discussion centering around the current formal specification of UML[1] structural and behavioral semantics [7] and formal visual specifications[2].

In the paper at hand we first describe the components of modeling methods and their degrees of formalization in Section 2. Based on these foundations we present a framework for analyzing the degree of formalization of enterprise modeling methods. We then apply the framework to six

[1] Unified Modeling Language (UML) Superstructure Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/, Last Access: 2013-08-29

[2] Diagram Definition (DD) Specification Version 1.0, http://www.omg.org/spec/DD/1.0/, Last Access: 2013-08-29

978-1-4799-2504-9/14 $31.00 © 2014 IEEE
DOI 10.1109/HICSS.2014.422

common enterprise modeling methods to identify the degree of formalization of their process-related aspects in Section 3. A discussion on how these insights influence the value of the resulting models for machine-based processing follows in Section 4. The paper is concluded with an outlook on further research steps and potential implications of the research for the conceptualization of modeling methods.

2. Foundations

In this Section we will outline the foundations for generally describing modeling methods. Therefore, we consider the modeling language, modeling procedure, and mechanisms and algorithms. Subsequently, we will discuss the formalization of the constituents of modeling methods. The goal is to describe, how to specify modeling methods formally. This results in a comparison framework which can then be applied in Section 3 in order to analyze six enterprise modeling methods.

2.1. Enterprise modeling methods

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, organization of an enterprise.

A central aspect of almost any enterprise modeling approach is the definition of the behavioral aspects of an enterprise by means of processes [8]. These processes are usually described using models. They play a vital role for the enterprise which is why the design of work systems, supporting and realizing those processes, should be strongly aligned to the process models.

Several authors have analyzed selected enterprise modeling methods based on a given application domain or usage scenario. Lakhoua et al. [9] analyze several enterprise modeling methods according to the domains they can be applied in, consistency, polyvalence, and simulation. Szegheo and Andersen [10] investigate enterprise modeling methods based on their underlying approach (i.e. active knowledge modeling, process modeling, object-oriented modeling, agent-based systems) and conclude, that every approach has its application area. The authors describe in which scenarios the individual approaches are most suitable. The literature up to now only considers investigating existing enterprise modeling methods according to their suitability for a given application domain or context. The paper at hand offers a different view on enterprise modeling methods by analyzing their degree of formalization in the specification of their process-related aspects.

As has been mentioned above, modeling methods permit to facilitate the management of complexity in the area of enterprise information systems. In order to characterize the constituents of modeling methods from a generic perspective we will revert to a framework developed by Karagiannis and Kühn in the following [11]. As highlighted in Figure 1, modeling methods in this framework are composed of two parts: a modeling technique and mechanisms and algorithms. A modeling technique is further composed of a modeling language, which includes the syntax, semantics, and notation, as well as a modeling procedure. The modeling procedure defines the steps and results for applying a modeling language. The mechanisms and algorithms are used in the modeling procedure based on the definition of the modeling language. Whereas algorithms stand for arbitrary steps of calculations that are applied to instances of a modeling language, mechanisms are lightweight functionalities implemented in a modeling tool that are targeted towards supporting users of the modeling language when engaging in modeling. They thus encompass aspects such as for example constraint checking of models, user interface aids for creating models, e.g. by providing specific dialogues to ease the creation of multiple elements simultaneously, or visualization functionalities that help a modeler to discover relationships between elements during modeling. Mechanisms and algorithms may either be generic, i.e. they can be applied to arbitrary modeling languages, specific, i.e. they are coupled to a specific modeling language, or hybrid, i.e. they can be parameterized for several modeling languages.

2.2. Formal aspects of modeling methods

In order to classify modeling methods according to their degree of formalization in Section 3, we now introduce a set of criteria for our investigation. The criteria are based on the generic concepts of modeling methods illustrated in Figure 1 by means of a UML class diagram (the components are visualized as classes, connected by composition, generalization, and association relationships). We analyze for each of the central components (i.e. Modeling Language, Modeling Procedure, Mechanisms & Algorithms), if

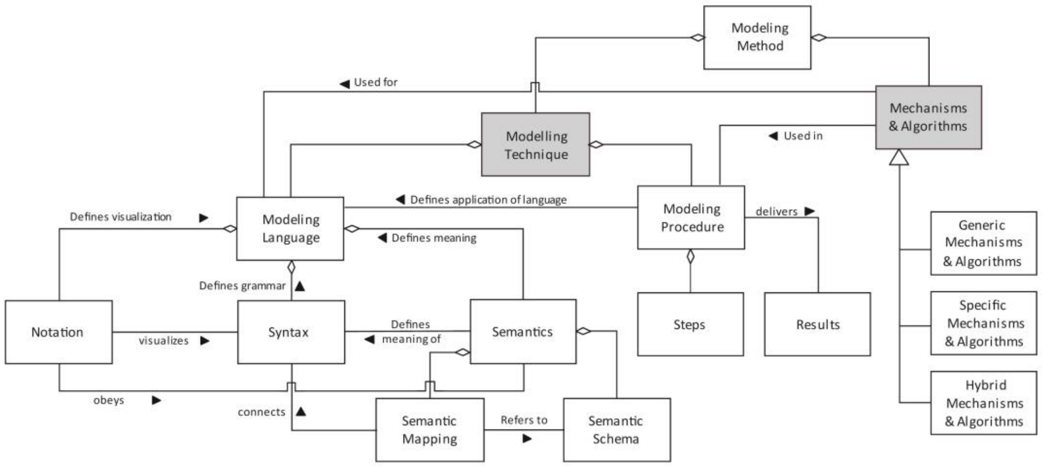

Figure 1. Component of modeling methods [11]

and how they can be formalized. In the context of this paper we define a ”formal definition” as one that provides an unambiguous specification that is intersubjectively understandable and processable by different computer systems.

2.2.1. Modeling language. A modeling language's constituents are *notation*, *syntax*, and *semantics*. 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 (e.g. FDMM [12], Z [13]).

The formal specification must define the elements as well as the set of possible relations between those elements by means of cardinalities. An informal specification of syntax is e.g. the definition of elements using natural language. Semi-formal specifications result from the combination of formal and informal specifications, i.e. some elements are specified formally but some others are introduced in natural language. Notation and semantics of a modeling language must be investigated in more detail as they relate to both, structural and behavioral aspects of the process models. Formal semantics assigns an unambiguous meaning to each element of the language's syntax [14].

In our study, we decompose notation into *static notation* and *dynamic notation*. If the notation of a language's element is fixed at all time, we refer to a static notation, if the notation can change depending on the current state (i.e. attribute value) of the element, we refer to a dynamic notation. Generally, the range of specifications for notations span from purely informal (e.g. using natural language) over semi-formal (e.g. defining shapes: ellipse, rectangle) up to formal (e.g. shapes implemented with a programming language, or a precise mathematical description).

For the semantics, structural and behavioral aspects are investigated individually. In our analysis, structural semantics is decomposed into *type semantics* and *inherent semantics* as introduced by Höfferer [15]. Type semantics is usually defined with the meta model of a modeling method by providing semantics for each model element (i.e. type) on a meta level. Inherent semantics describes the semantics of concrete instances of the meta model elements (i.e. concrete instances in the model). Both can be defined formally using e.g. an ontology as ontologies ”include computer-usable definitions of basic concepts in the domain and relationships among them” [16]. The behavioral semantics describes the degree of formalization according to the process model execution. A formal specification of the behavior can be provided by e.g. relating to the Petri net semantics or by providing some algebraic definition. Both, structural and behavioral semantics can be informally described by using natural language. The semantic domain ”specifies the very concepts that exist in the universe of discourse” [14]. Its description can be in natural language (i.e. informal) or rigorous mathematics (i.e. formal). In order to provide a formal semantic mapping, a rigorously defined function from the language's syntax to its semantic domain can be defined [14].

2.2.2. Modeling procedure. The modeling procedure is defined by *steps* and *results* in Figure 1. These criteria describe, how the user actually builds models, i.e. the sequence of actions performed by the modeler in order to create valid models. A formal specification of a modeling procedure can be provided e.g. by using rule-based systems, triple-graph grammars, or constraint definition languages. Informally, a description of the sequence of steps in e.g. a tabular manner can be given. Semi-formal specifications would combine formal approaches for some steps while remaining on an informal level for other ones.

2.2.3. 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 for the notation, the development of approaches for a formal specification of mechanisms and algorithms specifically for modeling methods is still an open research area. Mechanisms and algorithms are e.g. simulation algorithms that can be executed on models or model transformation algorithms, transforming a source model into a target model by providing a mapping function between the source meta model and the target meta model [5]. A formal representation in this field can be stated, if meta model mappings are defined or concrete algorithms are given in a programming language. An informal description can again be in natural language, whereas using pseudo code notation or some language constructs together with natural language could be classified as semi-formal.

Figure 2 visualizes the analysis framework applied during the following investigation.

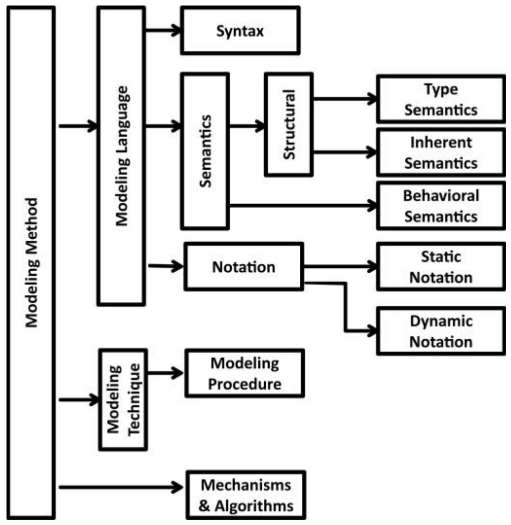

Figure 2. Analysis framework

3. Analysis of selected enterprise modeling methods

In the following section we analyze a set of enterprise modeling methods based on the formalization of the process-related aspects of the methods. The selection of methods is based on the involvement of the authors in the implementation and ample experience with three particular methods (i.e. BPMS, HORUS, SOM), contrasted with an international standard (i.e. UML) and one of the most widely used methods (i.e. ARIS), as well as one method with unique formal characteristics (i.e. TOVE). In our analysis we refer to the comparison criteria in Section 2.2.

3.1. ARIS framework

The architecture of integrated information systems (ARIS) framework, first published in 1992 by August- Wilhelm Scheer [17, 18], introduces an integrated framework for describing an enterprise. 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) [19], developed by the University of Saarland together with the SAP AG in 1992. EPCs are widely used in industry and still part of SAP process modeling components [20]. Our analysis concentrates on the process view of the framework.

Central concepts of an EPC are *function*, *event*, and *connectors*. 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 are used to define the control flow. These concepts are syntactically described using a meta model, their type semantics is described informally using natural language. An inherent semantics is not defined. The behavioral semantics of EPCs is initially described informally. Additional research aligned the behavioral semantics to Petri net theory [21]. Therefore, the behavioral semantics can be stated as formal. EPCs utilize the simulation of its instances. The static notation of function, event, and connector is semi-formally defined by 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 into a more comprehensive formal specification of the semantics of EPCs (e.g. [22]), especially considering the non-local semantics (e.g. [23]). The publications state, that the formal aspects of the Petri net semantics is not sufficient to describe the behavioral semantics of EPCs appropriately. In order to overcome that shortcoming several approaches are 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” [24], however providing their own formalization some years later [25]. Semantic domain and semantic mapping are defined on an informal level only. As the initial publication of EPCs defines some decomposition guidelines for functions and events, an informal specification of the modeling procedure is available.

3.2. BPMS method

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” [26]. The method has been developed at the University of Vienna. 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 a wide range of industrial projects [27].

BPMS integrates three abstraction levels, *business level*, *execution level*, and *evaluation level*. Each is defined by one or more dedicated *BPMS-Processes* (i.e. strategic decision, reengineering, resource allocation, workflow, and performance evaluation). The creation of business process models is assigned to the *reengineering process*. Therefore we will investigate the reengineering process in the following.

The syntax of the business process modeling component is described formally using an algebraic notation [26]. The type semantics of the central modeling elements (i.e. activities, subprocesses, and control flow) is described in natural language. Sample simulation algorithms are informally defined [28] and implemented in the Adonis BPMS modeling tool. Some algorithms have currently been specified semi-formally [5]. Herbst [29] introduced a mapping from the 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 inter-dependencies of the several process phases, no modeling procedure 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 for some elements of the method, but not for the relations [29]. A dynamic notation is not defined.

3.3. HORUS

Horus is an enterprise modeling method that focuses on business process engineering and includes steps for the integrated modeling of business processes, the improvement of business processes, and the application of the created models [30]. For these purposes, the Horus method comprises four phases for: preparing process optimization projects, elaborating the strategy and architecture, analyzing business processes, and applying the results. To investigate the formalization dimensions of the process-related aspects in Horus, we will restrict our analysis to the third phase of Horus. At the core of this phase stand so-called *procedure models* that can be further linked to organization 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. In XML nets the objects in the places are XML documents and transitions are operations on these XML documents using XQuery[3] statements.

For the syntax of Horus procedure models a mathematical specification is available based on the FDMM formalism [31]. 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 [32] can be characterized as formal. Accordingly, also the semantic mappings and the semantic domain for procedure models are formally defined. For inherent semantics, Horus procedure models currently do not offer any facilities. The behavioral semantics of procedure models are also formally defined by inheriting the operational semantics of Petri nets. The static notation of procedure models is described semi-formally through graphical illustrations. In addition, procedure models also feature dynamic notation, e.g. for depicting organizational resources that are linked to transitions in [32, p. 54ff.]. These dynamic aspects are also illustrated informally. For the modeling procedure Schoenthaler et al. show detailed diagrams based on a Petri-net-like notation on how to use the various model types. However, this is not detailed down to the level of modeling objects. Therefore, we classify the modeling procedure as semi-formal. Horus procedure models can be used for simulation algorithms. These are explained by Schoenthaler et al. in a semi-formal style with mainly textually describing their behavior together with examples for illustrating corresponding calculations.

[3] See http://www.w3.org/TR/xquery/ accessed 2013-08-29

3.4. SOM

The Semantic Object Model (SOM) [33] method is a multi-perspective enterprise modeling method. SOM compromises a layered approach for a comprehensive description of an enterprise with an enterprise plan on the top layer, a business process model on the second layer, and the specification of resources on the third layer. An emphasis of SOM is on the specification of business process models using a multi-view approach [34]. SOM business process models integrate structural and behavioral aspects, modeled in different views, into one comprehensive model. SOM defines views for the structural aspects (i.e. the *Interaction Scheme*), the behavior (i.e. the *Task-Event Scheme*), and the decomposition of business objects and business transaction in *Decomposition Diagrams*. In the following we refer to the business process modeling part of SOM [35].

The syntax of SOM business process models is described in a meta-model. The meta-model also includes the static notation of the model elements by providing their respective shapes (e.g. environmental objects as ellipses, business objects as rectangles, business transactions as arrows). A dynamic notation for SOM is not defined. The type semantics of the business process models is described informally using natural language, but referring to the concepts of systems theory, transaction-based coordination, and object-orientation. The behavioral semantics on the other hand 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 [33]. Semantic mapping as well as semantic domain are only described on an informal level.

The modeling procedure is described with precise decomposition rules that can be applied to business transactions and business objects respectively. The rules are described formally using a Backus-Naur notation. They specify how modelers can apply them recursively to detail and refine an initial business process model thereby revealing the coordination between the business objects. Modeling mechanisms and algorithms, considering the business process modeling part, are not defined for SOM.

3.5. TOVE

The diversity of enterprise models and legacy systems supporting the creation of those models has led to the *correspondence problem* [36]. This means, that it is hard if even possible to compare different enterprise models. The authors behind TOVE therefore introduce the idea of a *General Enterprise Model* (GEM), that can be extended to a concrete domain, therefore defining a *Deductive Enterprise Model* (DEM). GEMs consists of three parts: A taxonomy of object classes, for each object class, relations to other object classes plus a definition of the semantics of the relation, and a set of attributes for each object class, together with the intended meaning of the attribute [36]. Fox et al. used this GEM to develop the *Toronto Virtual Enterprise Deductive Enterprise Model* (TOVE), developed at the University of Toronto. 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 in order to enable automatic, deductive answering of common questions on the enterprise, and defining graphical symbols for the components [37]. In our analysis we will concentrate on the *Activity-State Model* which is based on the *Activity-State Ontology* of TOVE.

The syntax of the Activity-State model can be stated to be semi-formal, because on the one side, all elements of the model are described comprehensively using taxonomies, on the other side a formal specification according to their connectivity is missing [38]. The semantics according to structure and behavior is completely defined formally, meaning, that not just the type semantics is described using ontologies but also the inherent semantics. The activity-state-time ontology uses the *situation calculus* [39] as foundational theory for describing the semantics to the ontologies of activity, state, and time [36]. Therefore, the behavioral semantics are formally defined. The static notation of the elements are defined semi-formally by providing some sample models, whereas the dynamic notation is not defined. The semantic domain is formally defined using ontologies describing the semantics of the object classes and relations, whereas the semantic mapping is on a semi-formal level as it is defined on the semi-formal syntax. Fox et al. informally describe a concrete sequence of actions, a modeler must undertake in order to create a TOVE model. For an Activity-State model itself is no modeling procedure defined. As the model is directed towards a deductive competence, the questions - defined in first-order logic and programmed in Prolog - represent some mechanisms & algorithms of TOVE in a formal way.

3.6. UML

The Unified Modeling Language (UML) is the de-facto industry standard for the object-oriented specification of software-intense systems [40]. 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 structural aspects and activity or sequence diagrams for behavioral aspects). The UML is used on a broad basis today. Besides the already defined set of diagrams by the OMG, modelers have the possibility to create UML profiles. These UML profiles enable the UML to be applicable in new and emergent domains. Nevertheless, we concentrate on behavioral aspects in our analysis and therefore investigate the *UML Activity Diagrams* in the following.

Activity Diagrams (AD) are used to specify the dynamic behavior of the modeled system by defining activities, and relations between activities. The central elements of an AD (i.e. activities and activity edges) are formally introduced using a meta model[4] derived by the Meta-Object Facility (MOF) meta meta model[5]. Activity diagrams utilize Petri net like semantics. Therefore, the behavioral semantics of ADs can be stated as formal. Considering the structural semantics, UML provides a semi-formal type semantics for the elements of ADs. Besides the informal description of the semantics using natural language, each diagram element is related to a concept of the meta model, and therefore to a concept of the MOF, by a generalization relation. An inherent semantics is not defined in the specification. 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 the semantic mapping and semantic domain no specifications have been defined. Although the integrated MOF meta meta model allows for a hands-on definition of meta model mappings for the transformation of ADs into related diagrams, generally no mechanisms and algorithms are defined. The same is true for the modeling procedure, as the UML generally doesn't define any procedure of how models should be created.

Although we use the standard specification for our analysis, we want to show some very interesting current research topics aligned to the UML and extensions directed to formal specifications. Engels et al. [40] introduce a semantic domain and a semantic mapping in order to utilize concrete consistency tests between different UML diagrams. Others introduce a formal semantics by mapping the AD concepts into a mathematical expression called *Activity Calculus* [41]. The OMG itself is also working on introducing formal specifications by defining a *Diagram Definition* language[6] (DD). 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.

Another interesting development direction according to the UML is the *Foundational UML* or *fUML* specification[7]. With fUML, the Object Management Group defines an execution semantics for a subset of the UML constructs. The goal is to ”enable compliant models to be transformed into various executable forms for verification, integration, and deployment”[7]. Accordingly, conformance can be evaluated on two aspects: (1) *syntactic conformance*, a conforming model must be restricted to the abstract syntax, and (2) *semantic conformance*, a conforming execution tool must provide execution semantics for a conforming model that is consistent with the semantics defined for fUML.

4. Discussion

The analysis shows that the six enterprise modeling methods investigated in this paper differ largely according to their degree of formalization in the process-related aspects. According to our analysis framework, neither one method is completely specified formally, nor completely informally. TOVE is the only enterprise modeling method providing a formally defined inherent semantics for the created models, whereas HORUS is the only method providing at least an informal specification for the dynamic notation. With five of ten criteria defined formally and three more criteria defined on a semi-formal level, TOVE and HORUS are the methods with the highest degree of formalization in our study. On the other side of the spectrum is the UML, having in the current version only two criteria formally defined. However, according to the UML, several extensions to the current standard introduce more formal specifications (see Section 3.6). Table 1 sums up the results gained during our investigation.

Introducing formalization enables the resulting models to be intersubjectively understandable (i.e. unambiguous) and processable by different computer systems. Automated verification, validation, and model testing is also a positive aspect of formal

[4] Unified Modeling Language (UML) Superstructure Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/, Last Access: 2013-08-29

[5] Meta Object Facility (MOF) Core Specification Version 2.4.1, http://www.omg.org/spec/MOF/2.4.1/PDF/, Last Access: 2013-08-29

[6] Diagram Definition (DD) Specification Version 1.0, http://www.omg.org/spec/DD/1.0/, Last Access: 2013-08-29

[7] fUML Specification Version 1.0, http://www.omg.org/spec/FUML/1.0, Last Access: 2013-08-29

	ARIS	**BPMS**	**HORUS**	**SOM**	**TOVE**	**UML**
Syntax	formal	formal	formal	formal	semi-formal	formal
Semantics						
Structural Semantics						
Type Semantics	formal	informal	formal	informal	formal	semi-formal
Inherent Semantics	n/a	n/a	n/a	n/a	formal	n/a
Behavioral Semantics	formal	formal	formal	formal	formal	formal
Notation						
Static Notation	semi-formal	semi-formal	semi-formal	semi-formal	semi-formal	semi-formal
Dynamic Notation	n/a	n/a	informal	n/a	n/a	n/a
Semantic Mapping	informal	formal	formal	informal	semi-formal	informal
Semantic Domain	informal	formal	formal	informal	formal	informal
Modeling Procedure	n/a	n/a	semi-formal	formal	n/a	n/a
Mechanisms & Algorithms	formal	semi-formal	semi-formal	n/a	formal	n/a

Figure 3. Results of the analysis of the process-related aspects of six enterprise modeling methods

specifications. Generally, introducing a formal specification enables testing model conformity on a deeper level.

Whereas most enterprise modeling methods allow for syntactic conformity checking (i.e. whether the elements in the model conform to the abstract syntax defined in the meta model), some others already provide a semantic conformity checking (i.e. whether the semantics of a model element is consistent).

Accordingly, modeling tools can be separated into (1) tools, providing syntactic conformity, and (2) tools that also provide a semantic conformity. Models created with a formally specified semantics can be compared also on a semantic basis because the subjectivity of models is reduced, therefore providing an intersubjective understanding of the model. As TOVE is the only method in our analysis providing a completely formal specification of its semantics (i.e., type, inherent, and behavioral semantics), TOVE models can be compared semantically and fully automated. The definition of a formal behavioral semantics enables the interchange e.g. between a modeling tool and a simulation tool seamlessly (e.g. for BPMS models).

If modeling procedure and mechanisms and algorithms are specified on a formal level (e.g. in the case of SOM), a tool developer can use these specifications as a requirements specification for building a conforming modeling tool. Additionally verification & validation of the implementation can be checked.

The formal specification of the behavioral semantics – e.g. for HORUS through the mapping to Petri-nets - directly enables the re-use of existing analysis functionalities e.g. simulation approaches for Petri-nets.

To sum up we can state, that the level of interoperability depends on the level of formalization. To an extreme, using the diagram definition approach of the UML, one is able to provide conformity not only on syntactic and semantic level but also on notational level.

Besides the discussed positive effects, introducing more formal specifications also reduces the degree of freedom for modelers and "increases the effort involved in creating the specifications" [6]. As modeling is 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 modeling method and the dedicated model users individually. Current developments around the UML – e.g. fUML, diagram definition - on the other hand indicate, that more aspects of modeling methods should be specified in a formal way. "However, to enable meaningful exchange of model information between tools, agreement on semantics and notation is required"[8]. This is particular important for advanced knowledge systems to leverage additional value of models.

5. Conclusion

Mastering the complexity during the design and development of work and knowledge systems one can refer to enterprise modeling methods and enterprise models. These models can be created using a rich set of diverse modeling methods. In order to provide an unambiguous understanding of those models and to foster the interoperability between different computer systems, the introduction of

[8] Unified Modeling Language (UML) Infrastructure Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/, Last Access: 2013-08-29

formal specifications is a necessary step towards managing the heterogeneity and complexity. In addition, the models provide value themselves by acting as knowledge bases.

The paper at hand investigated six common enterprise modeling methods based on their degree of formalization regarding the process-related aspects. First, a general comparison framework has been introduced. This framework has then been applied to the modeling methods resulting in a comprehensive discussion of the benefits and drawbacks of formalization in modeling method specification.

In the future, we are planning to apply the introduced framework to an even broader set of modeling methods by reverting to the Open Models Initiative [42] (OMI). The OMI provides a platform for tool developers and researchers for conceptualizing and implementing modeling methods. This will also include the extension of the framework to cover non process-related aspects.

References

[1] S. Alter, "Defining information systems as work systems: implications for the IS field", European Journal of Information Systems, vol. 17, pp. 448–469, 2008.

[2] R. Maier, Knowledge Management Systems: Information and Communication Technologies for Knowledge Management. Springer, 2004, Second Edition.

[3] Y. Wand and R. Weber, "Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda", Information Systems Research, vol. 13, no. 4, pp. 363-376, 2002.

[4] A. Maes and G. Poels, "Evaluating quality of conceptual modelling scripts based on user perceptions", Data & Knowledge Engineering, 63(3):701 - 724, 2007.

[5] H.-G. Fill and D. Karagiannis, "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform", Enterprise Modelling and Information Systems Architecture, vol. 8, no. 1, pp. 4-25, 2013.

[6] P. Meseguer and A. D. Preece, "Assessing the Role of Formal Specifications in Verification and Validation of Knowledge-Based Systems", in Proceedings of the Third International Conference on Achieving Quality in Software, pp. 317–328, Chapman & Hall, 1996.

[7] E. Seidewitz, "UML: Once More with Meaning", 2013, talk at the University of Maryland, 2013/04/15. [Online]. Available: http://www.isr.umd.edu/sites/default/files/Seidewitz041513.pptxg, Last Access: 2013-08-28

[8] H. Shen, B. Wall, M. Zaremba, Y. Chen, and J. Browne, "Integration of business modelling methods for enterprise information system analysis and user requirements gathering", Computers in Industry, vol. 54, no. 3, pp. 307 - 323, 2004.

[9] M. Lakhoua and M. Rahmouni, "Investigation of the methods of enterprise modeling", African Journal of Business Management, vol. 5, no. 16, pp. 6845-6852, 2011.

[10] O. Szegheo and B. Andersen, "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.

[11] D. Karagiannis and H. Kühn, Metamodeling Platforms. Proceedings of the Third international EC-Web 2002 – Dexa, Aix-en-Provence, Springer, 2002, p. 182.

[12] H.-G. Fill, T. Redmond, and D. Karagiannis, "FDMM: A Formalism for Describing ADOxx Meta Models and Models", in Proceeding of ICEIS 2012 - 14th International Conference on Enterprise Information Systems 2012, vol. 3, pp. 133–144.

[13] J. Abrial and O. U. C. Laboratory, Specification Language Z: Basic Library. Oxford University Computing Laboratory, 1980.

[14] D. Harel and B. Rumpe, "Meaningful Modeling: What's the Semantics of "Semantics"?", Computer, vol. 37, no. 10, pp. 64–72, Oct. 2004.

[15] P. Höfferer, "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.

[16] L. Obrst, "Ontologies for Semantically Interoperable Systems", in Proceedings of the twelfth International Conference on Information and Knowledge Management, ser. CIKM '03, 2003, pp. 366–369.

[17] A. W. Scheer, Architektur integrierter Informations-systeme: Grundlagen der Unternehmensmodellierung. Springer, 1992.

[18] A.-W. Scheer and K. Schneider, "ARIS Architecture of Integrated Information Systems", in Handbook on Architectures of Information Systems, P. Bernus, K. Mertins, and G. Schmidt, Eds. Springer Berlin Heidelberg, 2006, pp. 605–623.

[19] W. Hoffmann, J. Kirsch, and A. Scheer, Modellierung mit Ereignisgesteuerten Prozeßketten: Methodenhandbuch, Veröffentlichungen des Instituts für Wirtschaftsinformatik. 1992.

[20] G. Keller and T. Teufel, SAP R/3 prozessorientiert anwenden: iteratives Prozess-Prototyping zur Bildung von

Wertschöpfungsketten, ser. Edition SAP. Addison-Wesley-Longman, 1997.

[21] R. Chen and A.-W. Scheer, Modellierung von Prozessketten mittels Petri-Netz-Theorie. ser. Institut für Wirtschaftsinformatik im Institut für Empirische Wirtschaftsforschung an der Universität des Saarlandes, 1994, no. 107.

[22] M. Nüttgens and F. J. Rump, "Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)", in Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen - Promise 2002, LNI, J. Desel and M. Weske, Eds., vol. 21, 2002, pp. 64-77.

[23] E. Kindler, "On the Semantics of EPCs: A Framework for Resolving the Vicious Circle", in International Conference on Business Process Management (BPM 2004), Lecture Notes in Computer Science, 2004, pp. 82-97, Springer-Verlag.

[24] W. V. D. Aalst, J. Desel, and E. Kindler, "On the semantics of EPCs: A vicious circle", in Proceedings of the EPK 2002: Business Process Management Using EPCs, 2002, pp. 71–80.

[25] J. Mendling and W. V. D. Aalst, "Formalization and Verification of EPCs with OR-Joins Based on State and Context", in Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Springer-Verlag, 2007, pp. 439–453.

[26] D. Karagiannis, S. Junginger, and R. Strobl, "Introduction to Business Process Management Systems Concepts", in Business Process Modelling, B. Scholz-Reiterand E. Stickel, Eds. Springer Berlin Heidelberg, 1996, pp. 81–106.

[27] P. Harmon,"The BPTrends 2010 BPM Software Tools Report on BOC's Adonis Version 4.0", BPTrends, http://www.bptrends.com/publicationfiles/2010%20BPM%20Tools%20Report-BOCph.pdf, last checked: 2013-09-11. 2010.

[28] J. Herbst, S. Junginger, and H. Kühn, "Simulation in Financial Services with the Business Process Management System ADONIS", in Proceedings of the 9th European Simulation Symposium (ESS'97), 1997.

[29] J. Herbst, "Ein induktiver Anstaz zur Akquisition und Adaption von Workflow-Modellen", Ph.D. dissertation, University of Ulm, 2001.

[30] F. Schönthaler, G. Vossen, and A. Oberweis, Business Processes for Business Communities: Modeling Languages, Methods, Tools. Springer, 2012.

[31] H.-G. Fill, S. Hickl, D. Karagiannis, A. Oberweis, and A. Schoknecht, "A Formal Specification of the Horus Modeling Language Using FDMM", in Proceedings of the 11th International Conference on Wirtschaftsinformatik (WI2013), R. Alt and B. Franczyk, Eds. Merkur-Verlag, pp. 1165 – 1179, 2013.

[32] K. Lenz and A. Oberweis, "Interorganizational Business Process Management with XML Nets", in Petri Net Technology for Communication-Based Systems, Advances in Petri Nets, Springer-Verlag, 2003, vol. 2472, pp. 243–263.

[33] O. K. Ferstl and E. J. Sinz, Grundlagen der Wirtschaftsinformatik, 7th ed. München: Oldenbourg, 2013.

[34] D. Bork and E. J. Sinz, "Bridging the Gap from a Multi-View Modelling Method to the Design of a Multi-View Modelling Tool", Enterprise Modelling and Information Systems Architecture, in press, 2013.

[35] O. K. Ferstl and E. J. Sinz, "Modeling of Business Systems Using SOM", in Handbook on Architectures of Information Systems, P. Bernus, K. Mertins, and G. Schmidt, Eds. Berlin: Springer, 2005, pp. 347–367.

[36] M. S. Fox and M. Gruninger, "Enterprise Modeling", American Association for Artificial Intelligence, vol. 19, no. 3, p. 109, 1998.

[37] M. S. Fox, "The TOVE Project Towards a Common-Sense Model of the Enterprise", in Proceedings of the 5th international conference on Industrial and engineering applications of artificial intelligence and expert systems, ser. IEA/AIE '92, 1992, pp. 25-34.

[38] M. S. Fox, J. F. Chionglo, and F. G. Fadel, "A Common-Sense Model of the Enterprise", in Proceedings of the Industrial Engineering Research Conference, 1993, pp. 425–429.

[39] R. Reiter, "Artificial Intelligence and Mathematical Theory of Computation", V. Lifschitz, Ed., 1991, The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression, pp. 359–380.

[40] G. Engels, J. M. Küster, R. Heckel, and L. Groenewegen, "A Methodology for Specifying and Analyzing Consistency of Object-Oriented Behavioral Models", in Proceedings of the 8th European Software Engineering Conference, ACM, 2001, pp. 186–195.

[41] Y. Jarraya and M. Debbabi, "Formal Specification and Probabilistic Verification of SysML Activity Diagrams", in Sixth International Symposium on Theoretical Aspects of Software Engineering (TASE'2013), 2012, pp. 17–24.

[42] D. Karagiannis, W. Grossmann, and P. Höfferer, "Open Model Initiative - A Feasibility Study", 2008, last checked: 2013-08-30. Available: http://cms.dke.univie.ac.at/uploads/media/OpenModels Feasibility Study SEPT 2008.pdf