Enterprise Modelling and Information Systems Architectures
Vol. 13, No. 5 (2018). DOI:10.18417/emisa.13.5

Semantic Annotations of Enterprise Models for Supporting the Evolution of Model-Driven Organizations

Hans-Georg Fill*,[a]

[a] University of Bamberg / University of Vienna, An der Weberei 5, 96047 Bamberg, Germany

Abstract. *In model-driven organizations enterprise models are used to represent and analyze the current and future states of aspects such as strategies, business processes or the enterprise architecture. Thereby, the scope of representation and analysis depends on the used modeling method. Although adaptations of modeling methods are frequently conducted to meet requirements emerging from business changes, such modifications may not be favorable due to potential side effects on other enterprise systems, e. g. through inconsistencies with existing standards and resulting conflicts in algorithmic processing. In the paper at hand we therefore propose the use of semantic annotations of enterprise models for dynamically extending the representation and analysis scope of enterprise modeling methods. Through a loose-coupling between enterprise models and formal semantic schemata, additional information can be represented and processed by algorithms without changes in the original modeling language. In this way, the evolution of information requirements of an organization can be satisfied while maintaining the consistency of the used enterprise modeling languages. For illustrating the feasibility of the approach we describe a use case from the area of risk management. The use case is realized using the SeMFIS platform that supports the annotation of enterprise models and the subsequent machine-based analysis of annotations.*

Keywords. Semantic Annotation • Enterprise Models • Model Analysis

Communicated by T. Clark. Received 2016-06-28. Accepted after 1 revision on 2018-01-23.

1 Introduction

The adaptation of organizations to changing environmental conditions and new technologies in order to stay competitive has been a traditional function of management (Raineri 2011). A particular challenge for companies today is to keep pace with the large volume of ground-breaking technologies and especially the digital transformation of traditional business models (Hui 2014; Porter and Heppelmann 2015). In this context, the alignment of clear definitions of strategic visions and business cases with the effective use of information technology is a major concern (Fitzgerald et al. 2013). However, as has been found in a recent global survey by Bain & Company (Rigby and Bilodeau 2015), the complexity stemming in part from today's globally acting organizations and the involved massive amounts of communications often hinder innovation and growth.

As a solution for mastering complexity, the concept of *model-driven organizations* has been proposed (Clark et al. 2014; Groth 1999). In an early conception of model-driven organizations, conceptual models were positioned as a coordination mechanism for organizations that enable formal and precise descriptions of complex organizational and technical relationships (Groth 1999). Already at that time, the vision was formulated that organizations could be managed through *regulating models*, "where the controlling and automating aspect of information technology is exploited to the maximum" (Groth 1999, p. 17). In the work by Clark et al. (2014), the vision of

* Corresponding author.
E-mail. hans-georg.fill@uni-bamberg.de

making models the primary means for interacting with and for evolving the organization has been considerably detailed and related to the most recent developments in enterprise modeling and model-driven engineering, cf. (France and Rumpe 2007; Frank 2014a; Karagiannis et al. 2008). In particular, the interweaving of IT platforms and domain-specific models of organizational aspects has been promoted for enabling advanced analyses of the structure and behavior of organizations.

Although such a model-driven approach already greatly supports decision makers in handling the involved complexity, the fast evolution of organizations due to the above mentioned changes still persists. Thus, the used modeling approach has to be flexible and responsive to new requirements and resulting changes of the organization and its IT systems (Fill and Karagiannis 2013; Fox and Grüninger 1998). Thereby, special attention needs to be given to fundamental changes in requirements for domain-specific modeling languages (Frank 2011). This type of modeling languages is specifically tailored to the requirements and semantics of application domains and thus needs to co-evolve with the underlying domain (Chen et al. 2005). Due to this tight coupling with the domain, domain-specific languages also tend to change more frequently than general purpose languages (Wile 2001).

Consider for example that new requirements emerge due to changes in legal regulations that make it necessary to record information in a different way. The corresponding domain-specific modeling languages then have to be adapted for ensuring the compliance to the legal regulations by satisfying these information needs, cf. (Fill et al. 2007). If organizations widely adopt a model-driven approach – as it can already be witnessed in many large enterprises today and expected for the next years on a broader basis (Rosemann 2006; Sandkuhl et al. 2018) – changes in a modeling language may however not be easily accomplished. This stems from the fact that in model-driven organizations not only various human actors interact with the models. Similarly, also machines in the form of autonomous agents may access the models and the contained information for taking decisions (Blair et al. 2009). Changes in the underlying model schema can thus lead to conflicts in interpreting the model content.

In Fig. 1 the central aspects of changes in modeling languages are depicted. For the purpose of better comprehensibility, we show only the aspects relevant for our subsequent elaborations. A further discussion on the relationships of modeling languages, models, and their instantiation can be found in (Fill et al. 2012; Henderson-Sellers 2012). On the left hand side, the conformance relationship between a modeling language ML and the model instances created with this language M_n is shown. In addition, algorithms A_m have been added that process the model contents and that refer to the definitions given by the modeling language. As pointed out by Harel and Rumpe (2004), algorithms operating on models can be regarded as semantic mappings between a modeling language and appropriately represented system runs as a semantic domain. Through these mappings the behavioral semantics of modeling languages can be represented.

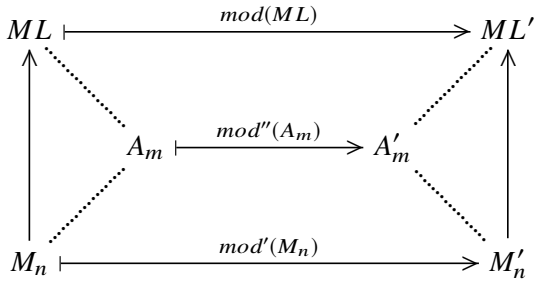

Figure 1: Effects Resulting from Changes in a Modeling Language

If new requirements emerge that necessitate changes in the modeling language, a modification of the modeling language $mod(ML)$ is conducted. This results in a new version of the modeling language depicted as ML'. For ensuring the correct instantiation of models from this new modeling language, the models may also have to be adapted.

In the literature this is typically denoted as *co-evolution of models* (Wachsmuth 2007). It is expressed by the adaptation $mod'(M_n)$, which leads to the modified set of models M'_n. As the algorithms equally depend on the definitions given by the modeling language, they may have to be adapted as well. This is expressed by $mod''(A_m)$, which leads to the modified set of algorithms A'_m.

The effort that results from the changes in the modeling language is hard to quantify. Estimations can be derived from recent work on metamodel evolution that illustrates the involved complexity (Jahn 2014; Khelladi et al. 2015). Thereby it is however primarily focused on changes in the modeling language together with some constraints or transformations – see e. g. (Demuth et al. 2013) – without taking into account full-fledged algorithms and only some kind of semantics, e. g. (Wachsmuth 2007). The overall effort thus depends not only on the type of changes applied to the modeling language, on the degree of coupling between the algorithms and the modeling language and the complexity of the algorithms themselves. It also needs to be taken into account if and what kind of domain knowledge is required for conducting the changes – i. e. adapting a generic serialization algorithm that does not require specific knowledge about the underlying domain will typically require less effort than adapting a domain-specific simulation algorithm (Fill and Karagiannis 2013)[p.16ff.]. It can however be stated for the worst case, that for every change in the modeling language each of the m algorithms has to be adapted and $m \times n$ tests have to be run on the models to verify their correctness.

To prevent such conflicts, model-driven organizations typically establish guidelines for ensuring the consistent use of modeling languages and for preventing arbitrary changes. As an example consider the Swiss guideline *eCH-0158*. This guideline lists the elements, relations, and attributes of BPMN models as they shall be used in public administration[1] . However, the question arises how organizations shall remain agile and adapt easily to new requirements if the modeling languages are restricted in such a way? And even if additional information may be encoded by using additional textual descriptions, how shall this information be effectively processed?

This leads us to the following research questions that we are going to investigate in the remainder of the paper. The first research question is directed towards the adaptation of modeling methods in general and modeling languages in particular. *R1:* Do approaches already exist for evolving modeling methods and modeling languages that do not violate potential modeling guidelines nor affect algorithms that interpret model content? The second research question takes an engineering perspective for adding an innovative contribution to this state-of-the-art. *R2:* How can an intuitive and visual approach be designed and technically realized that permits to extend the semantic representation and analysis scope of modeling languages without affecting the original modeling language? And finally, the third research question *R3:* How can the feasibility of such an approach be demonstrated? The paper is therefore organized as follows: In Sect. 2 we will briefly define some terminological foundations to ensure a common understanding of the terms used throughout the paper. In Sect. 3 related work on evolving modeling methods will be discussed to give answers to the first research question. Subsequently, in Sect. 4 an approach based on semantic annotations of enterprise models will be presented as an innovative contribution for answering research question two. To illustrate its feasibility, it will be applied to a use case from the area of risk management in Sect. 5 thereby answering research question three. Finally, the benefits and limitations of the approach will be discussed in Sect. 6. The paper will be concluded with an outlook on future work.

2 Terminological Foundations

In this section we will briefly define the core terminology that will be used in this paper. It shall ensure a common understanding across the many

[1] https://www.ech.ch/vechweb/page?p=dossier&documentNumber=eCH-0158 last accessed 04-02-2018

disciplines that are concerned with enterprise modeling such as business modeling, business process modeling, information modeling, enterprise engineering, or requirements engineering and management (Feldmann et al. 2014; Sandkuhl et al. 2014).

Regarding the concept of a *modeling method*, we revert to the terminology proposed by Karagiannis and Kühn (2002) – see also Fig. 2 that describes the relationships in diagrammatic form.

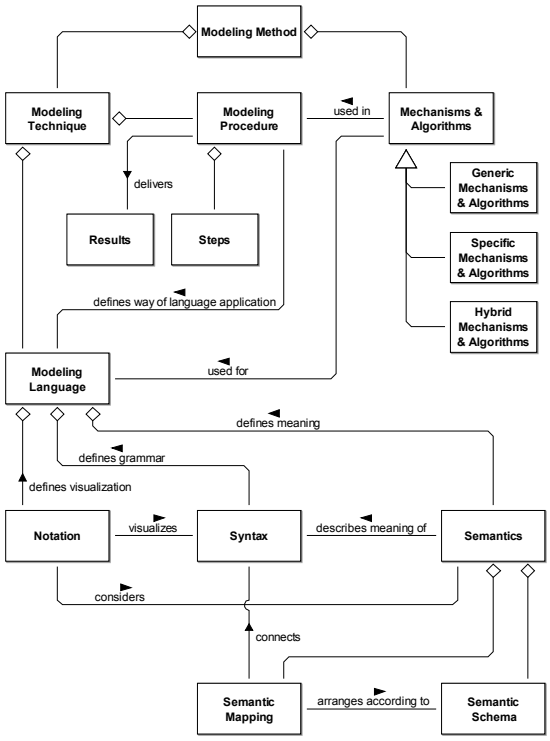

Figure 2: Used Terminology for Modeling Methods, based on Karagiannis and Kühn (2002)

According to this terminology, modeling methods are composed of a *Modeling Technique* and *Mechanisms and Algorithms*. The modeling technique is then divided into a *Modeling Language* and a *Modeling Procedure*. The modeling procedure relates to *methods* as discussed in the field of *method engineering*, which is concerned with system development processes in the context of information system development (Rossi et al. 2004). However, in the context of modeling methods, it is not a general procedure for system development but focuses on the *Steps* of applying a modeling language to deliver *Results*. The concept of the modeling language follows in large parts the definitions as found in the area of computer science (Harel and Rumpe 2000). Thereby, the *Syntax* defines the grammar of the modeling language and the *Semantics* defines the meaning of the elements and expressions of the grammar. In this context, the definition of the syntax of a modeling language is often denoted as the *metamodel*. The specification of semantics is accomplished by using a *Semantic Mapping* that connects the syntax to a *Semantic Schema*. Regarding the concept of a *Notation* we take a different view than Harel and Rumpe (2000) who use it synonymously with syntax. The notation in our terminology refers to the visualization of the modeling language (Fill 2009b; Moody 2009). To complete the characterization of a modeling method, the mechanisms and algorithms can be of three different types. *Generic Mechanisms and Algorithms* stand for such that can be used for arbitrary modeling languages. Examples for this type include generic layout algorithms or algorithms for the generic serialization of object-oriented model representations in XMI (Di Battista et al. 1999; OMG 2015b). The second type are *Specific Mechanisms and Algorithms*. These are tied to a specific modeling language and are thus primary candidates for adaptations if the modeling language changes. Examples include simulation algorithms for particular business process modeling languages such as token-game algorithms or discrete-event simulations (Desel 1997; Herbst et al. 1997). Finally, the third type are *Hybrid Mechanisms and Algorithms*. These are configurable so that they are applicable to multiple modeling languages. An example would be a simulation algorithm that can be configured for different types of process modeling languages (Fill and Karagiannis 2013).

In addition to the structure of modeling methods, each of the components of a modeling language can be further characterized regarding their degree of formality. Besides other aspects this is important for processing the information in models by

machines, for ensuring its unambiguous interpretation and thus for enabling interoperability. In this context, formal specifications are unambiguous specifications that are inter-subjectively understandable and processable by machines (Bork and Fill 2014). We can further distinguish between formal and semi-formal specifications (Fraser et al. 1994). In semi-formal specifications only some parts of the specification are expressed in an unambiguous manner, whereas other parts are described using ambiguous definitions such as natural language. The use of semi-formal specifications is common for many enterprise modeling languages where the syntax is formally described but the semantics is only given in natural language – see e. g. the various OMG specifications for BPMN and UML. In contrast, (fully) formal specifications define all elements in an unambiguous manner, e. g. using mathematical expressions (Tarski 1936).

3 Related Work on Evolving Enterprise Modeling Methods

In order to answer the first research question that has been posed in the introduction, we advance now with an investigation of previous related work on how to evolve enterprise modeling methods. Due to the specific requirements in the context of enterprise modeling (Atkinson et al. 2013), where changes in modeling languages may not be easily accomplished, we are particularly interested in approaches that do not require such modifications. Furthermore, our focus is not to remove existing information that is contained in models but rather to add information in a processable format without affecting existing processing capabilities (Braun 2015). Thus, we regard a sub-area of the more general discussion on the evolution of modeling languages, which includes all modifying operations on a modeling language, i. e. creation, deletion, updating (Demuth et al. 2013; Jahn 2014). This narrows our analysis to *non-detrimental extensions* of what we will denote as the *representation and analysis scope* of modeling methods.

Based on the constituents of modeling methods as described in the previous section, such extensions can be related to the syntax and semantics of a modeling language. Further, it can be found that extension capabilities may have already been conceived during the design of a modeling language, which we denote as *a-priori extensions*, or have to be added later, which we denote as *ex-post extensions*.

In the category of a-priori extensions of modeling languages, several enterprise modeling languages today offer features that explicitly target the addition of information. In the widely-used Business Process Model and Notation (BPMN), two options for extending the modeling language are offered. On the one hand additional, user-defined attributes can be added to BPMN elements that are not contained in the specification (OMG 2011)[p.57]. For this purpose so-called *BPMN extensions* have to be defined using the constructs *ExtensionDefinition*, *ExtensionAttributeDefinition*, and *ExtensionAttributeValue*. The additional attributes may include also the definition of a separate XML structure. Due to the integration in the standard specification and the restriction of adding attributes only, these extensions are still standard compliant. The second option for extending the scope of BPMN is provided by so-called *external relationships* (OMG 2011)[p.61]. Corresponding *Relationship* elements can reference metadata elements in Complete Meta Object Facility (CMOF) specifications and thus become part of the modeling language. These external artifacts can thus also be integrated in a compliant way in BPMN models without violating the BPMN specification. Nevertheless, potential modeling guidelines set forth by organizations may restrict the usage of such extensions – as e. g. imposed by previously mentioned Swiss guideline eCH-0158.

A similar approach for extensions can be found in the Unified Modeling Language (UML) (OMG 2015a). In UML this is accomplished via so-called *profiles* that permit the customization of the UML metamodel (OMG 2015a)[p.250]. This is accomplished via so-called stereotype constructs that can be used to re-define existing types in UML. In this

way, aspects such as changes in the terminology, in the notation or for adding semantics to UML classes can be realized. However, this mechanism is specific to UML.

Regarding ex-post extensions of modeling languages, several approaches have been inspired by concepts used in programming languages. De Lara and Guerra (2010) have identified three directions for specifying re-usable behavior for modeling languages that can be added ex-post. Thus, they do not target an extension of the representation scope but only the analysis scope, i. e. in terms of the processing by algorithms. This includes the use of: a.) *generic concepts in parametric types*, b.) *model templates*, or c.) *mixin layers*.

The first direction follows an approach that is common in object-oriented programming. In order to add behavior to classes, object-oriented programming approaches typically provide *inheritance* or *sub-typing* mechanisms. Thereby, behavior that has been specified for upper level classes is transferred to classes on a lower level of the modeling language. This approach has also been used in the area of modeling methods. In Fill and Karagiannis (2013) for example, behavior for simulation algorithms is transferred to user-defined metamodels by inheriting from a pre-defined metamodel whose concepts can be processed by an algorithm.

The approach of adding types to concepts in a modeling language can also be applied ex-post as has recently been shown by De Lara et al. (2015). Thereby, additional types are assigned to existing instances to re-classify objects. This view is also taken in the upcoming field of multi-level modeling or deep (metamodeling) (Atkinson et al. 2014). The central idea thereby is to permit the definition and modification of types in a modeling language by users during run-time, which are then used for the specification of instances as well[2] . Multi-level modeling may either be realized using specialized modeling infrastructures and tools or through configurations of traditional modeling tools (Gogolla 2015). A first application of this approach to industry scenarios in the area of enterprise architecture management showed that it needs to be combined with traditional modeling approaches as well as mixins to become feasible for enterprise modeling (Trojer et al. 2014). Despite the great potential of such typing approaches, their major drawback is their invasiveness. If more than one behavior specification shall be included, it needs to be inherited from several metamodels or multi-level models thus leading to complex inheritance hierarchies and the pollution of the original modeling language (e. g. through necessary specification of the potency, i. e. to control on which levels concepts may be instantiated) (Kolovos et al. 2010; Langer et al. 2012). These may not only be hard to understand by domain experts, which are the traditional target users of enterprise modeling approaches. The implementation of robust algorithms on such highly-dynamic information structures also seems challenging due to the large degree of freedom during run-time.

As a solution to this drawback, the use of *concepts* and *templates* / *mixin layers* has been proposed (De Lara and Guerra 2010). Concepts take a similar approach as the inheritance mechanisms by adding behavior through additional information structures. These can for example be other metamodels, models, statements in programming languages, etc. As an example consider the approach of dynamic metamodeling by Engels et al. (2000) that uses formal graph transformation rules for specifying the behavior of UML models. In contrast to the inheritance mechanisms, the direction of adding the behavior to an existing modeling language is however reversed. This means that the information structures required for example by algorithms are bound *to* a modeling language instead of inheriting *from* it. Thereby, the original modeling language is not modified. Concepts thus have to provide mechanisms to bind their information structures to a modeling language, which can subsequently be interpreted by an algorithm.

Templates or mixin layers build on the approach of concepts. Mixin layers have originally been

[2] For a detailed characterization of multi-level modeling approaches we refer to (Atkinson et al. 2014)

introduced in the area of object-oriented programming languages (Smaragdakis and Batory 2002). They are a mechanism to eventually inherit from a superclass and provide functionality for already existing classes. In this way, additional functionality can be added to classes inheriting from a particular superclass. In the context of modeling languages, mixins act as components of the modeling language that provide certain behavior via information structures and that can be integrated through loosely bound concepts (De Lara and Guerra 2010).

A further kind of approaches that is similar to the concept-based approach but that focuses specifically on using other models as reference targets is discussed under the term *model weaving*. Model weaving has been defined as a "generic operation that establishes correspondences with semantic meaning between complex model elements" (Del Fabro et al. 2005)[p.4]. It originated from the area of model transformation where weaving models have been used to specify in detail the mapping between different model types. The main aspect that distinguishes model weaving from the approach of concepts is the presence of a specific *weaving model* which defines references to the source and the target models. Model weaving can therefore not only be used for defining the behavior when processing models. It also permits to add other information structures to existing models without modifying the original modeling language or the models.

Finally, there exist also approaches that propose more lightweight extensions of modeling languages and models and that may also be suitable for enterprise modeling. In Langer et al. (2012), an approach for lightweight profiles for the Eclipse Modeling Framework (EMF) is presented. It takes a similar direction as described above for UML profiles but leverages some of the restrictions on UML profiles. For example, it permits to add more complex data structures and gives an exact interpretation of the involved semantics for the profile extensions. With the approach of *model decorations* it is reverted to the annotation facilities provided in EMF (Kolovos et al. 2010). By adding *EAnnotation* elements to EMF metamodels, additional information is inserted into existing modeling language definitions that can be used for example for model transformations. Subsequently, this information is removed from the metamodels to avoid their pollution and replaced by the operations defined through the annotations, e. g. for injecting additional information or processing the model content. Again, this approach is similar to mechanisms in programming languages, where annotations are used to dynamically inject additional behavior (Krahn and Rumpe 2006). In the area of enterprise modeling, annotations have been used in the past to make knowledge that is contained in models explicit and thus machine-processable. This has been taken up for example in semantic business process management where annotations were used to discover web services, which could then execute annotated process activities (Born et al. 2007; Lin 2008). These approaches however either transfer all model information first to an ontology, e. g. (Born et al. 2007; Lin 2008) or extend the set of attributes of an existing modeling language to permit the addition of annotations, e. g. (Fill 2009a).

In summary, it can be concluded that the following research results have been achieved so far: the provision of mechanisms in modeling methods for extending their representation and analysis scope during the creation of models; the ex-post representation of additional information in models and the ex-post addition of behavior specifications via concepts, templates, mixins, and decorations; and the addition of information via model weaving. In some of the previous contributions the specific requirements of enterprise modeling methods have already been addressed. However, in line with the findings by Trojer et al. (2014) it can be found that previous approaches have mostly focused on modeling methods in the area of software technology. This applies in particular to extensions focusing on the semantics and behavior of models, which is an essential part for conducting simulations or machine-based analyses of enterprise models. As most of the currently available approaches require profound technical knowledge on the design of

modeling languages, they are not immediately usable by domain experts in business disciplines. At the same time, the extensions need to be directly machine-processable. Therefore, simple textual annotations are not sufficient. Furthermore, not all of the approaches give detailed control over the extensions, e. g. to store and process extensions of standard-based modeling methods separately. Little attention has so far been given to using open standards for specifying extensions, e. g. to enable interoperability across platforms and disciplines.

4 Semantic Annotations of Enterprise Models

Based on the findings on existing approaches we will present in this section an approach for extending the semantic representation and analysis scope of enterprise modeling methods by using so-called *semantic annotations*. It will be shown how the approach can be technically realized. Thereby we give an answer to research question R2 and lay the foundation for subsequently showing the feasibility of the approach.

4.1 Concept of Semantic Annotations

As we have already discussed in Sect. 3, annotations have been introduced in the area of programming languages to add behavior to existing code fragments. Through mechanisms that are able to interpret the annotations and the content that is annotated with them, behavior can be injected into programs. The approach of model decorations by Kolovos et al. (2010) showed how the same principles can be applied to the area of modeling. However, the drawback there was that the annotations polluted the original metamodel which may have unwanted side-effects and requires access to the metamodel by the user who wants to add annotations.

For avoiding this, we revert to a distinct modeling language for describing the annotations that has initially been proposed in Fill (2011a). This language permits to define separate annotation models that reference elements in a source model and a target model. It can thus be regarded as an approach of model weaving in the sense of Del Fabro et al. (2005). The use of an annotation model also gives greater control over the extensions of a modeling language than it would be possible using profiles or templates. Not only can such annotations or parts thereof be re-used for multiple purposes. Annotations further permit to add information structures *and* behavior. This can be accomplished through annotation-aware algorithms that can operate on annotated model content.

In contrast to the approaches that have been discussed so far, we use ontologies as target models for the annotations. One of the advantages of using ontologies is that they are today typically described using international standards in languages such as OWL or RDF (Horrocks et al. 2003; Obrst 2003). In the case of OWL for example, both the syntax and semantics of the ontology language are formally specified which leads to an unambiguous interpretation by humans and machines (W3C 2012). This is the basis for enabling interoperability and exchanging information across platforms and disciplines. Another aspect is that powerful tools are available for managing and processing ontologies (Gennari et al. 2003; Musen 2015), which contributes to their wide adoption in many areas of science and practice.

The relationships for the concept of semantic annotations are described by Fig. 3. On the left hand side it is shown how the original relationships between the modeling language ML, the models $M_{1..n}$, and the algorithms $A_{1..m}$ are maintained. On the right hand side the ontology language OL and the corresponding ontologies $O_{1..k}$ as instances of that language are depicted. In our approach, the annotations $Annot_{1..w}$ are then positioned on the model level and the level of ontology language instances. They are described by $Annot(M_i, O_j)_{1..w}$, thus linking a model M_i and and ontology language instance O_j. Above the annotations, the annotation-aware algorithms $A^O_{1..q}$ are shown.

For processing annotated model information, the algorithms therefore need to take into account

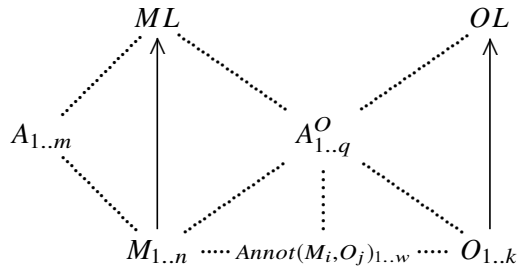

Figure 3: Concept of Semantic Annotations of Enterprise Models

not only the modeling language and the annotations but also the information expressed in the ontology language instances and potentially also the underlying ontology language. This adds complexity to the design of such algorithms. On the other hand, this approach permits to maintain the existing algorithms $A_{1..m}$ as well as existing models $M_{1..n}$.

The positioning of the annotations on the level of models and ontology language instances has been done for the purpose of usability by domain experts in business domains. The assumption thereby is that the annotation of modeling languages, which feature sometimes high technical complexity, is not feasible for the average business user. Although it may make sense from a technical viewpoint, e. g. in terms of simplicity for annotating all instances of a certain modeling class, the same effect can also be achieved on the level of models by providing according mechanisms in tools. In the subsequent section we will discuss this in more detail. Although we will regard in the following only the case of traditional two-level modeling environments for our approach, the concept of semantic annotation could also be applied to multi-level modeling approaches. Especially in domains such as enterprise architecture management and multi-view enterprise visualizations where considerable potential has been identified for multi-level enterprise modeling approaches (Atkinson et al. 2013; Trojer et al. 2014), it may be beneficial to specify the behavior of multi-level models via annotations as well. In that case the modeling language ML has to be regarded as the fundamental modeling infrastructure and the models $M_{1..n}$ as representations based on that infrastructure.

4.2 Model-based Semantic Annotation Using SeMFIS

The realization of the concept of semantic annotations of enterprise models can be accomplished in different ways. From the perspective of web information systems where annotations have been proposed in various formats to realize the vision of a semantic web (Berners-Lee et al. 2001), the most obvious approach would be to add the semantic annotations in some web standards-conforming format such as XML, e. g. (Chebotko et al. 2007). For this purpose, the models that shall be annotated would have to be serialized in such a format and in some cases transformed to instances of an ontology language. In this way however, the model environment, which is the primary working space of domain experts engaging in enterprise modeling, would be left. Therefore it has been decided to conduct the annotations within the modeling environment by providing a model-based annotation approach.

The approach that has been developed for this purpose is denoted as the *semantic-based modeling framework* or *SeMFIS* for short. SeMFIS is based on the conception of a distinct modeling language for semantic annotations as described in Fill (2011a). In addition to this annotation modeling language, SeMFIS provides model-based representations of ontologies. Beside the support for the widely-used web ontology language OWL, a model type for ontologies based on the OKBC frames ontology language as implemented in the Protégé toolkit (Gennari et al. 2003), as well as a simple term model type for controlled vocabularies are included. The modeling language of SeMFIS is shown by the metamodel in Fig. 4.

The *semantic annotation model type* on the left provides the concept for linking arbitrary elements in enterprise models to instances in one of the ontology languages. This is accomplished by the *model reference* and *connector reference*

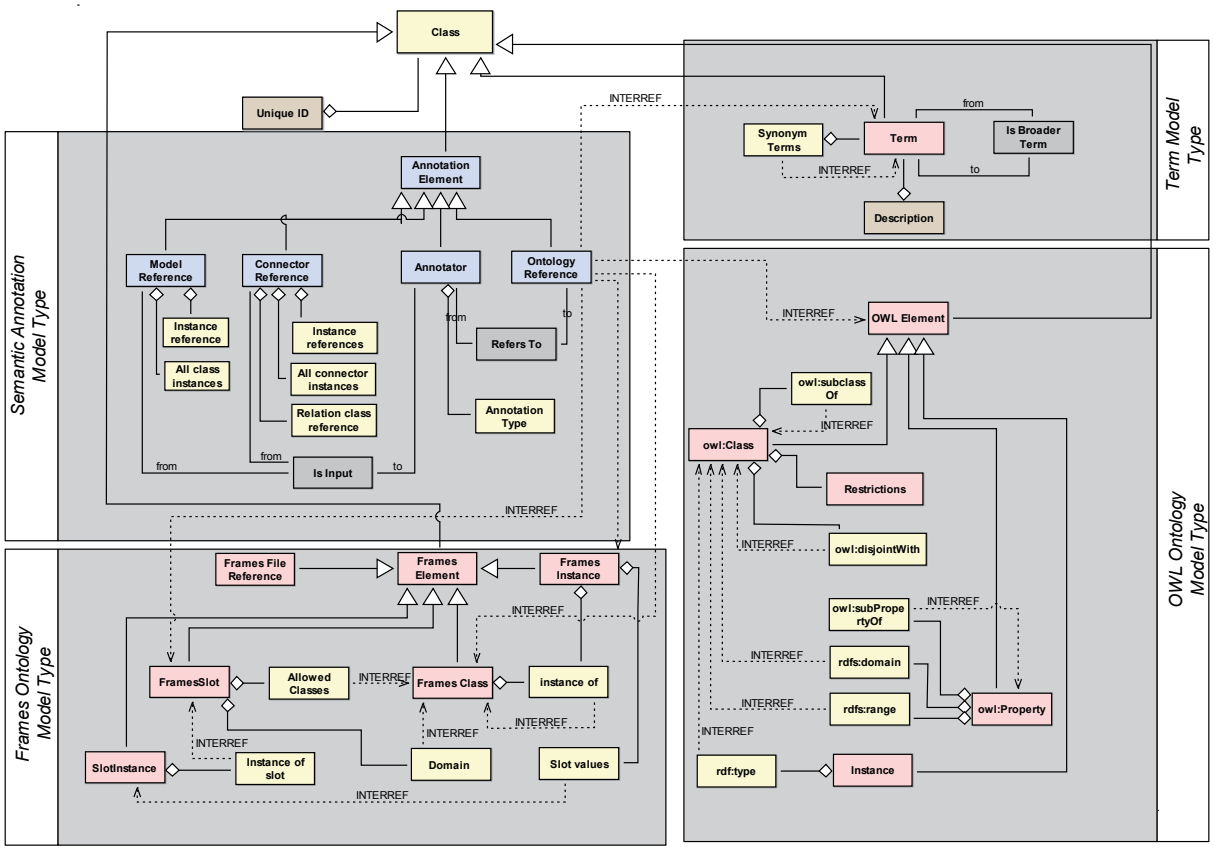

Figure 4: Excerpt SeMFIS Metamodel (Fill 2017)

elements, which reference to elements in the enterprise models. For linking to the ontology concepts, the *ontology reference* element is available. To connect these two references and the *annotator* element, the relations *is input* and *Refers to* are used. Thus, annotations are defined as triples consisting of a model or a connector reference, an annotator, and an ontology reference, as well as *Is input* and *Refers to* relations between these elements.

To ease the annotation of multiple elements in models, the model reference and the connector reference elements provide the *All class instances* attribute. By reverting to this attribute, it can be expressed that all instances of a certain model or relationclass shall be comprised by the annotation. The connector reference elements further provide the attributes *Relationclass reference* and *Instance references*. With the first attribute, the type of relationclass that shall be annotated is selected. Subsequently, based on this decision, the concrete instances of the selected relationclass can be added using the *Instance references* attribute. These two mechanisms, for annotating multiple instances and for annotating relationclass instances, have been designed with regard to business domain experts. Therefore, it is not necessary to have a technical understanding of the nature of modeling languages and metamodels - e. g. regarding potential inheritance hierarchies that would need to be taken into account – while still being able to express annotations for all elements of an enterprise model.

Concerning the choice of ontology languages, this was driven by the following factors. First, it should be reverted to established standards that enable interoperability. For this purpose the web ontology language OWL was chosen. Second,

despite its wide adoption, OWL comes with a considerable complexity that hampers its use in many business scenarios. Especially, the open world assumption that is the basis for OWL requires familiarity with the underlying description logic and needs to be considered during the design and use of ontologies (Horrocks et al. 2003). Therefore, it was chosen to add the *Frames Ontology Model* that is based on a closed world assumption and thus easier to comprehend for many people. Third, it may not be necessary for all applications of semantic annotations to use full fledged ontologies but rather remain on the lower formal spectrum (Obrst 2003). For this reason, the *Term Model* type has been included. It permits the definition of controlled vocabularies including synonym and superordination relationships.

The concept of the semantic annotation modeling language of SeMFIS does not depend on any particular metamodeling framework nor a particular enterprise modeling language. Although its design was influenced by the concepts of the ADOxx metamodeling approach – as will be shown in the next section – its fundamental concepts can be realized with any of the major metamodeling frameworks. The advantage of this kind of annotation language is seen in the absence of a coupling with a modeling language due to its design as a weaving model. It neither pollutes existing enterprise modeling languages nor does it require adaptations on the side of the used ontology languages.

4.3 Technical Realization

It is today available as a stand-alone desktop tool and provided as an open-source platform via the OMiLAB initiative[3] (Fill 2017). In the current release configuration of the SeMFIS platform, several enterprise modeling languages have been added to showcase the annotation approach and ease the usage of the approach Currently, this includes model types for BPMN models, UML class diagrams, Process Maps, Document models, as well as ADONIS BPMS process and working environment models (Fill 2017).

The central components of the architecture of the SeMFIS platform are shown in Fig. 5. On the left side, the components that have been re-used from the ADOxx platform are depicted. This includes the user interaction, the application components, and the repository.

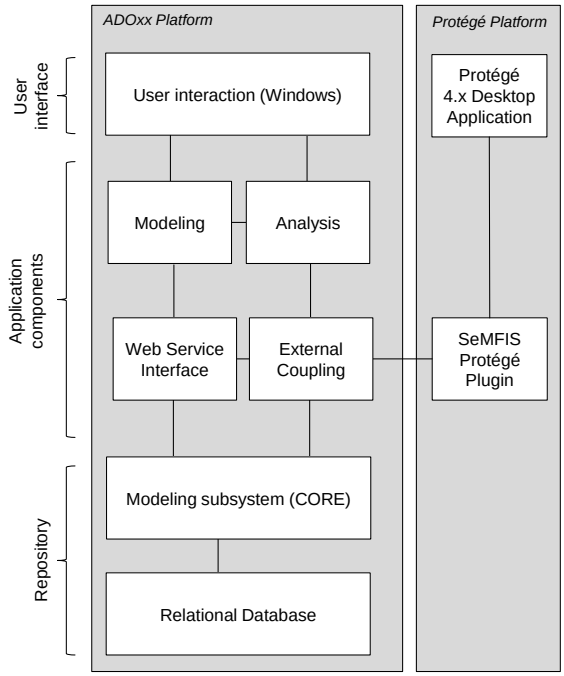

Figure 5: Central Components of the SeMFIS Architecture, condensed from Fill (2017)

As ADOxx is a Windows-based client-server framework, also the SeMFIS platform is a Windows application. The *modeling* component thereby automatically creates model editors from the metamodel specifications. Via the *analysis* component queries in the proprietary ADOxx query language (AQL) can be run on models. The *web service interface* provides a SOAP endpoint that can be used to integrate the platform in service-oriented architectures and access its functionalities and models over the web (Fill et al. 2013). The *external coupling* component offers the proprietary ADOscript scripting language that allows programmatic access to the platform. Based on

[3] See http://semfis-platform.org/ last accessed 04-02-2018

Dimension	**Semantic Annotations**	**Model Weaving**	**Semantic Mixins**
Targeted Modeling Level	Model Instances	Metamodels	Metamodels / Model Instances
Format of Additional Representations	Ontology Models	Arbitrary Types of Models	Arbitrary Types of Models
Notation	Visual	Tree View	Textual
Pragmatic Orientation	Human & Machine Processing	Transformation	Addition of Behavior
Underlying Execution language	ADOscript / Any via XML	ATL, XSLT	Java, Epsilon Object Language (EOL)
Scope of Join Operators	Pre-defined + user-defined	Equivalence and Calculation expressions	Everything expressable through sub-typing of concepts
Platforms with Available Implementation	SeMFIS (ADOxx)	ATLAS Model Weaver	Meta-Depth

Table 1: Comparison of Semantic Annotations, Model Weaving, and Semantic Mixins

this language, XML import and export facilities are realized that permit to exchange model information in a generic XML format. The *modeling subsystem* of the ADOxx platform automatically stores all model and metamodel information in a relational database.

In addition to the ADOxx components, a plugin for the Protégé ontology management platform is available for SeMFIS. It is used to serialize OWL ontologies that are stored in Protégé in a SeMFIS-compatible XML format. Thereby, OWL ontologies can be transferred from Protégé to the SeMFIS environment and used for annotation tasks.

4.4 Comparison with Existing Approaches

In summary, the approach of semantic annotations of enterprise models as realized in SeMFIS can be compared to two previous approaches that have taken similar directions for achieving non-intrusive extensions of modeling methods. This concerns in particular the approach of model weaving by Del Fabro et al. (2005) and the approach of semantic mixins by De Lara and Guerra (2010). For conducting a detailed comparison to these approaches, we use the following seven dimensions as shown in Tab. 1: *targeted modeling level*, *format of additional representations*, *notation*, *pragmatic orientation*, *underlying execution language*, *scope of join operators*, and *platforms with available implementation*.

Thereby, the targeted modeling level distinguishes whether an approach is directed towards the level of the modeling language (metamodels) or the instances of the modeling language. In the case of semantic annotations as described in this paper, the target are model instances whereas model weaving acts on the metamodel level to join different types of models. The approach of semantic mixins considers both levels.

With the approach of semantic annotations, the format of the additional representations is restricted to ontology models, whereas model weaving and semantic mixins permit to use arbitrary types of models. As shown above, semantic annotations as used in SeMFIS provide a separate visual modeling language for specifying the

linkage between conceptual enterprise models and ontologies. On the other hand, model weaving as implemented in the ATLAS platform provides a tree-based widget to define linkages. Semantic mixins are specified in a textual notation.

With the pragmatic orientation the goal of the regarded approach is expressed. Semantic annotations are suitable both for humans and machines to provide additional information for processing. Model weaving as described by Del Fabro et al. (2005) primarily targets tasks in model transformation. Semantic mixins have been described for adding behavior to modeling languages.

For processing the additional information by machines, the approach of semantic annotations on the SeMFIS platform provides the ADOscript language as well as arbitrary types of processing via its XML interface. By model weaving, the processing in the form of transformations is performed using ATL and XSLT. The approach of semantic mixins relies on the Java programming language and the Epsilon Object Language, that is used for manipulating EMF models.

When joining different information structures, it is essential to know about the type and scope of available join operators. For semantic annotations, there are some pre-defined join operators such as standard semantic reference operators (e. g. the synonym relationship). In addition, users can specify their own join operators as well. In model weaving, there are equivalence operators to express the equality of concepts as well as several calculation expressions for defining more complex joins. For semantic mixins, the expressiveness of the information to be joined depends on the type of information included in the concepts from which it is being sub-typed.

Finally, as has already been mentioned, the approach of semantic annotations has been implemented on the SeMFIS platform, which is based on the ADOxx metamodeling platform. The model weaving approach is provided by the ATLAS model weaver tool and semantic mixins have been added to the Meta-Depth platform for multi-level modeling.

5 Use Case: Risk Management

For illustrating the feasibility of the presented approach, we will describe in this section a use case. In this way we give an answer to research question R3. For the use case we selected the area of risk management that has been and still is a central task in today's organizations (Bromiley et al. 2015; Gericke et al. 2009). Risk management in enterprises today not only includes traditional risks such as liability risks and potential accidents, but stretches even to strategic risks such as product obsolescence and competitor actions (Bromiley et al. 2015). Besides the inherent interest of companies to successfully manage their risks, also the compliance with regulations of national and supra-national authorities and government bodies require organizations to know about their risks and handle them appropriately (Faisst and Buhl 2005; Fill et al. 2007). In the past, approaches have thus been developed that integrate aspects of risks with Business Process Management in order to describe the ways how risk management in conducted and to consider risks in the (re-)design of business processes (Suriadi et al. 2014).

For achieving this integration, a number of proposals have been made for extending existing business process modeling languages with risk concepts, e. g. (Fill et al. 2007; Weiss and Winkelmann 2011; Zur Muehlen and Rosemann 2005). In addition, domain-specific modeling languages and simulation approaches have been suggested, e. g. (Strecker et al. 2011; Tjoa et al. 2011). Although these approaches give answers to how risks can be represented and analyzed in business processes, they do not take into account how existing process models can be analyzed in regard to risk aspects without having to change the underlying modeling language. This is of particular concern as the widely used process modeling languages such as BPMN or EPC do not offer any risk-related information structures. Apart from risk aspects, there may be other compliance requirements issued by government authorities in the future that cannot be met with today's conceptions of enterprise modeling languages. However, such

requirements may still need to be documented and analyzed by companies. For this purpose, it has been suggested in Fill (2012a) to use semantic annotations of business process models to represent and process risks without having to change the underlying modeling languages.

In the following, we will take up and extend this approach for illustrating the application of semantic annotations to enterprise models. In contrast to the design presented earlier, we will build here on the BPMN modeling language and OWL ontologies as the two dominant standards today in business process management and ontologies. We thus start with the description of the involved modeling languages. As shown in Fig. 6, the basis for our discussion is an excerpt of the BPMN metamodel as contained in the ADOxx BPMN implementation[4] on the upper left of the figure.

For focusing on the semantic annotation approach, we only show six elements of the BPMN modeling language, i. e.: *Start Event*, *Task*, *Exclusive Gateway (XOR)*, *Parallel Gateway (AND)*, *Inclusive Gateway (OR)*, and *Event Gateway (Event-based Gateway)*. In addition, the relationclass *Sequence Flow* is shown that can be used to connect the aforementioned elements. On the bottom of Fig. 6, an excerpt of the OWL Ontology metamodel as provided by SeMFIS is depicted. It contains the central elements of the modeling language for representing OWL ontologies, i. e. *owl:Class* and *owl:Property*, as well as the attached attributes for connecting these entities, e. g. *rdfs:domain* and *rdfs:range* for defining the domain and range of properties or *owl:subClassOf* for expressing subsumption hierarchies of OWL classes. Furthermore, the modeling class *Instance* is part of the metamodel. With this class, instances of ontology classes can be represented. For this purpose, a reference relationship (INTERRREF) of type *rdf:type* is shown as well as a reference relationship of the type *Property*.

The third metamodel on the upper right shows the excerpt of the semantic annotation modeling language that is required for conducting annotations. Via the *Model Reference* class, reference relationships to any sub type of the *Process Element* class in a BPMN model can be established. Similarly, using the *Ontology Reference* class, reference relationships to all sub types of the *OWL Element* class can be defined. The references are then joined into an annotation triple with the depicted *Is Input* and *Refers To* relationclasses.

From these metamodel specifications, it already becomes visible that the semantic annotation approach does not lead to any changes or other kinds of pollutions in the BPMN modeling language. Rather, the same kind of referencing from the semantic annotation metamodel could be performed equally for any other enterprise modeling language.

Based on these specifications of the modeling languages, we can now advance to the level of model instances. For engaging in process-aware risk management, one of the first steps is to identify potential risks in a business process. Subsequently, these risks can be quantified in terms of impact and probability of occurrence. This information can then act as input for further analyses, e. g. to conduct calculations of the risk exposure of an enterprise or simulations of the effects of potential changes in risks and their occurrence in business processes. The basis for the identification of risks in business processes is the availability of a business process model.

In the upper left of Fig. 7, an excerpt of such a business process model using the BPMN modeling language is shown. It contains process information about the steps involved in electronically filing a tax declaration to the tax authorities. As the electronic filing of tax declarations is today a standard procedure in many enterprises, it serves well for illustrating the use case. In detail, Fig. 7 contains the BPMN *Task* instance *Submission*, which is

[4] The executable ADOxx BPMN implementation is provided for free here: https://www.adoxx.org/live/bpmn (last access 04-02-2018). In addition to the BPMN specification it features simulation functionalities for executing discrete-event-based path and capacity simulations. For this reason, some of the class and relationclass names have been adapted to fit with the ADOxx simulation configuration, see (Fill and Karagiannis 2013).

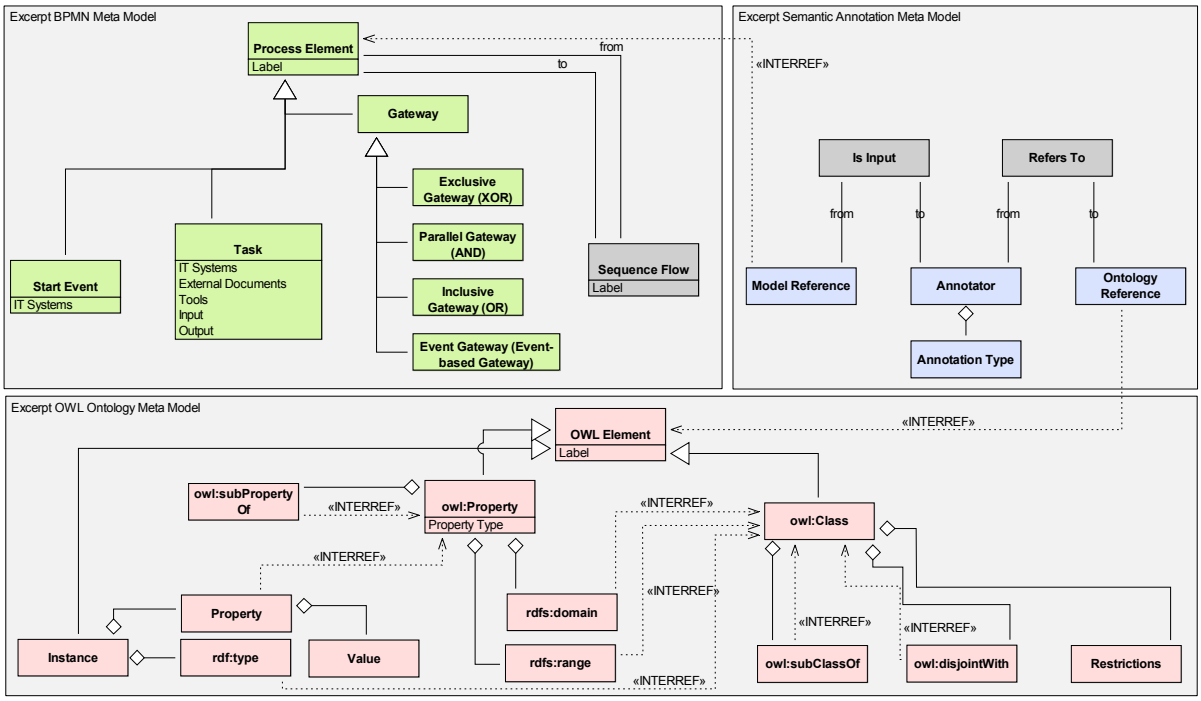

Figure 6: Metamodels for the Use Case showing Excerpts of the ADOxx BPMN Metamodel, the Semantic Annotation Metamodel, and the OWL Ontology Metamodel

followed via an instance of a *Sequence Flow* relationclass that leads to an instance of an *Exclusive Gateway (XOR)* named *Log-Out Decision*. These elements thus show the central steps in filing the tax declaration. As is is commonly known, the filing of tax declaration is subject to strict deadlines. If these deadlines are not met, there may be penalties for late payments, which may be substantial for large companies. It is therefore essential that all tasks in the filing process are completed and that potential risks hampering completion are identified and treated appropriately.

When aiming to identify risks, it can be reverted to publicly available risk catalogs in addition to individual elaborations. These catalogs are used to build on already accumulated knowledge on potential risks and thus enhance the quality of the risk identification. An example are the threat catalogs (IT-Grundschutz-Kataloge / Gefährdungskataloge) issued by the German Federal Office for Information Security (BSI)[5] . These threat catalogs contain a detailed hierarchy of potential threats regarding IT systems. They are classified into the five main categories *elementary threats*, *force majeure*, *organizational shortcomings*, *human errors*, *technical failures*, and *deliberate acts*. In each of these main categories, further sub-categories are defined, thereby leading to a detailed categorization for risks. For our use case, we reverted to these main categories and formalized them using an OWL ontology. As shown in Fig. 7 on the bottom, the excerpt of the OWL ontology contains an owl:class *GenericRisk* from which the two disjoint sub-classes *ElementaryThreats* and *TechnicalFailure* are defined. Certainly, this is only a small excerpt of the total ontology which could be easily further extended by using more of the described BSI risk categories. However, for the purpose of illustrating the core principles

[5] See https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/itgrundschutzkataloge_node.html (last access 03-06-2016).

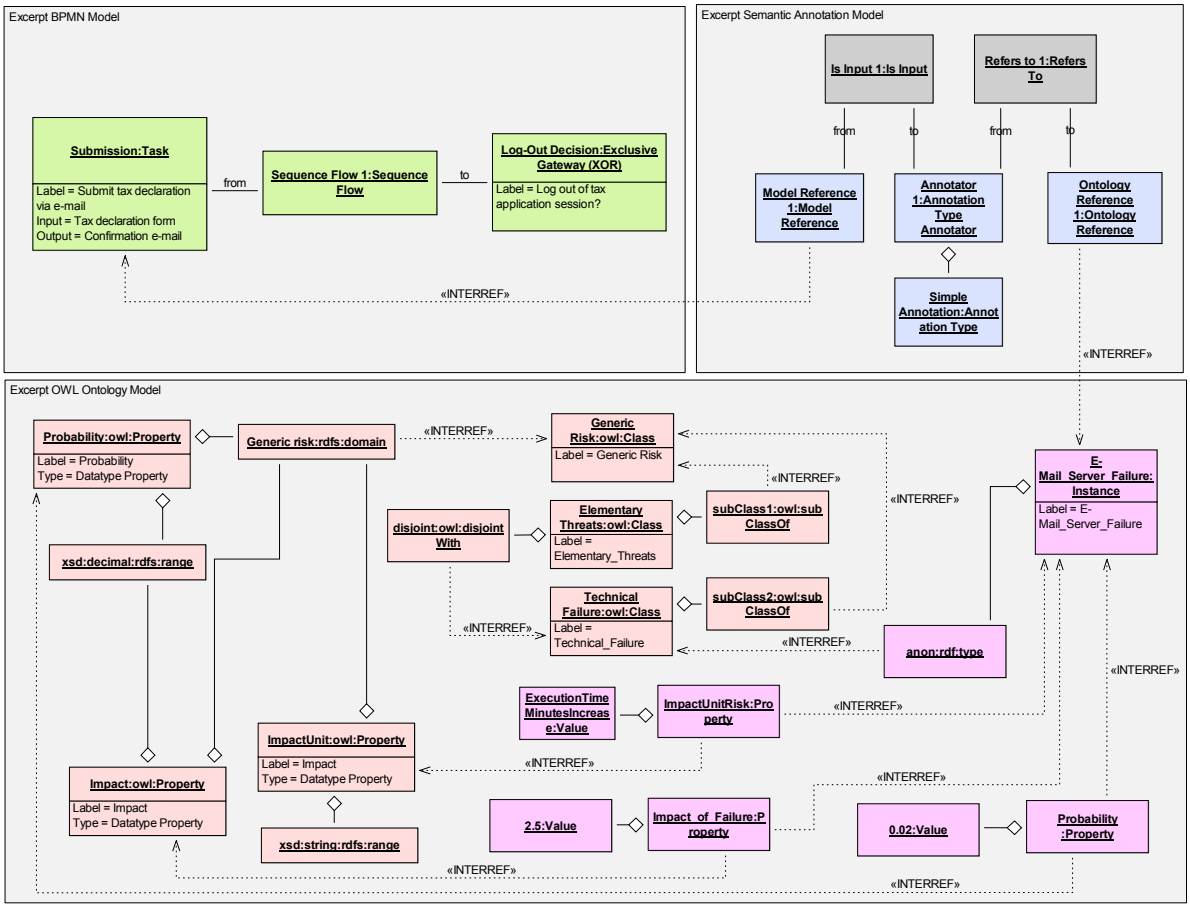

Figure 7: Models for the Use Case showing Excerpts of a BPMN Business Process Model, a Semantic Anntotation Model, and an OWL Ontology Model

this small sample will be sufficient. In addition to the class definitions, three properties have been defined: *Probability* for expressing the probability of occurrence for risks, *Impact* for defining the potential impact when a risk occurs, and *ImpactUnit* for defining the measurement unit of the specified impact. The *domain* of all three properties has been set to the *GenericRisk* owl:class, so that all sub-classes inherit them accordingly. In addition, for each property an appropriate *range* definition has been added. With these definitions in OWL, it can now be advanced to the definition of instances of this ontology.

In Fig. 7 an instance of the owl:class *TechnicalFailure* is shown. It is named *E-Mail_Server_Failure* and linked to the *TechnicalFailure* via an anonymous *rdf:type* reference relationship. Again, the direct linkage to the technical failure category of the BSI catalog is a simplification as it could be much more detailed in which of the many sub-categories of the BSI catalog it could be assigned to. Furthermore, property values are given for the instance, i. e. the *ImpactUnitRisk* instance relates to the *ImpactUnitProperty* and defines *ExecutionTimeMinutesIncrease* as the measurement unit for the impact property. The actual impact value is set by the instance *Impact_of_Failure* to the value *2.5*. The value of the instance *Probability* for the risk is set to the value *0.02*. With these figures and definitions, the risk can be quantified so that for example simulations and calculations could be applied,

cf. (Fill 2012a; Tjoa et al. 2011). As the risk ontology and the instances are specified in OWL, this information could also be easily exchanged with third parties and other platforms. Thereby the re-use of the information about the risks as expressed in the ontology is enabled, including its formal specifications.

The last step is to establish the linkage between the risks and the business process activities. Here, the semantic annotation model comes into play as shown in the upper right corner of Fig. 7. For expressing that the process task *Submit tax declaration via e-mail* is subject to the potential risk of an *E-mail server failure*, the BPMN task element is referenced by the semantic model via the *Model_Reference1* object and linked to the *Ontology_Reference1* object via the *Annotator1* object and corresponding relationships. The type of the annotator is thereby set to *SimpleAnnotation* as no additional information shall be conveyed by the annotation. The *Ontology_Reference1* in turn references the aforementioned OWL instance *E-Mail_Server_Failure*. In this way, it is formally expressed that the process task is annotated with the risk instance from the OWL ontology including its quantitative descriptions.

The next step is the technical realization of the use case on the SeMFIS platform and the machine-based processing of the such encoded information. As shown by the screenshot in Fig. 8, the above described models have been created using the SeMFIS model editors. In the chosen visualization, the references between the models are not shown. Neither are the sub:class references in the OWL model visualized, however they are of course present in the model specifications. The underlying reason for this is that the ontology information is rather used for searching for suitable annotation candidates here, which is typically easier accomplished using indented lists than graphs (Fu et al. 2017).

For illustrating how the additional information provided through the semantic annotations can be processed, we describe in the following a query for analyzing these relationships. For the specification of the query we revert to the ADOxx Query Language (AQL) that is also available within the SeMFIS platform. AQL is executable on SeMFIS and provides constructs for analyzing models, instances, relations, and attributes[6] . Although AQL is not yet as powerful as SQL in the world of databases, it is a mature and comprehensive query language for models. The queries can either a. be composed manually, b. by using a dialog-based editor or c. pre-defined for users in the form of templates. In the latter case, users can define values in these templates and execute the queries without further knowledge about AQL.

The query shown in code example 1 is assumed to be issued on the three models shown above. The statement in line 1 specifies the set of instances of the *Ontology reference* class that shall be retrieved by the query at first.

```
< "Ontology reference">
 [? "Ontology reference" = "
  REF mt: \"Ontology Model\"
        m: \"Risk Ontology OWL\"
        c: \"Instance\"
        i: \"E-Mail_Server_Failure\" "]
<- "Refers to"
<- "Is input"
- -> "Instance reference" >"Task"<
```

Code Example 1: AQL Query for Retrieving All Instances of the BPMN Task Class that are Annotated With the Risk Instance E-Mail_Server_Failure

This set is subsequently restricted to only those instances, where the attribute *Ontology reference* contains a reference to the OWL instance *E-Mail_Server_Failure* in the OWL model *Risk Ontology OWL* (lines 2-6). In the next part of the query, it is searched for all instances that are connected to any of these *Ontology reference* instances via the relation *Refers to* (line 7). Similarly, it is then further continued from these instances

[6] The documentation of AQL is publicly available on the ADOxx website: https://www.adoxx.org/live/adoxx-query-language-aql (last accessed 04-02-2018).

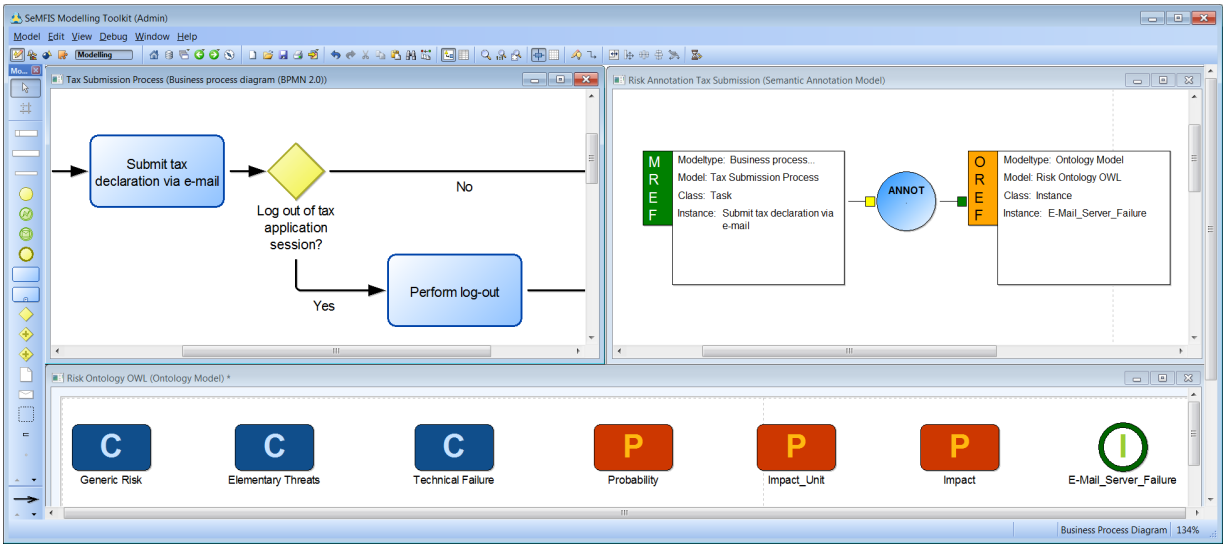

Figure 8: Screenshot of the Use Case Models in SeMFIS

to all instances that are connected via the *Is input* relation (line 8). Finally, from this last set, which only comprises any instances of the *Model reference* class, the contents of the *Instance reference* reference attribute are retrieved. These now contain all instances in models that have been used in the annotation. These are then restricted to such that are instances of the class *Task*.

The purpose of this query is thus to retrieve all tasks in a BPMN business process model that have been annotated using the *E-Mail_Server_Failure* risk. It could therefore serve the purpose of identifying potential improvements in a business process based on potential technical risks. Furthermore, additional information could be retrieved in the same way, e. g. for determining the details about the involved risks such as the probability of occurrence, the impact unit and the impact value. In line with the semantic annotation concept described in Fig. 3, the algorithm represented by this query has to be aware of the concepts of the original modeling language – in the use case this is BPMN, the model instances of this modeling language for finding the target elements, the information provided by the semantic annotation model, i. e. the triples consisting of the model references, the relations, and the ontology references, as well as of the used ontology language and its instances, i. e. the OWL instance concept. Although this is a rather simple example of a query, it contains all these relevant aspects. Certainly, more complex operations could be realized in the same way, e. g. by taking into account the semantics of the OWL ontology for retrieving all annotations related to instances of any technical failure or force majeure risks, for processing this information using simulation algorithms or for applying rule-based analyses based on the ontology – see (Fill 2012a) for further examples.

6 Discussion

With the descriptions given above, we can now discuss the advantages and limitations of the proposed approach. Along the answers provided to the three research questions posed in the beginning we conclude that several approaches have been identified that are able to evolve modeling methods without violating potential modeling guidelines nor affecting existing algorithms that interpret model content. Most of the approaches that we have investigated for this purpose stemmed mainly from the area of software technology and software engineering where similar requirements for evolving modeling methods exist. However, as

has been pointed out, the field of enterprise modeling with its particular focus on business experts who shall interact with the models, features some specificities not found in the area of software technology. Besides potential restrictions on the usage and scope of modeling languages, this concerns especially the way how information is encoded through models and how the user interaction takes place.

The concept of semantic annotation is regarded as a suitable solution for these aspects. Similar to other approaches in the area of model weaving and the concepts discussed by De Lara et al., semantic annotations do not require changes of the modeling language. At the same time, they permit the formal representation of additional information of arbitrary complexity. This is in contrast to the approaches of profiles, which have to be provided by a modeling language itself and may not leave the decision to the user what kind of information can be added. Furthermore, their use needs to be permitted by potential guidelines in an enterprise environment. The usage of ontologies of different levels of formality permits to choose a suitable formal level that is adequate for a given application environment. By building on the widely established OWL standard, ontology information can be easily exchanged with third parties and re-used in different contexts. With the availability of semantic annotations together with platforms such as SeMFIS, the interaction with them is thereby greatly eased. It can thus be expected that they are also suitable for domain experts who are not familiar with the underlying technical specifications. Of course, this would need to be tested in practice and the approach itself would have to be further evolved in order to reach the level of a professionally usable software application. However, due to the large effort involved in professional software development, this is typically not achievable by research prototypes in academic settings.

When taking the viewpoint of the vision of model-driven organizations, the concept of semantic annotations contributes to an enhanced agility and offers a means to quickly incorporate new information requirements in models. This is considered as a major advantage for organizations in terms of fast changing environments and resulting business challenges. In this way, semantic annotations are a possibility for fast adaptations, whereas the design of a new modeling language and the backloading and transitioning of existing model information to the new modeling infrastructure may require more time and greater caution. The applications of semantic annotations for model-driven organizations are thus manifold. Previous applications have shown their suitability for tasks such as business process benchmarking (Fill 2011b), the semantic obfuscation of model information for supporting information sharing (Fill 2012b), or the derivation of user-specific model visualizations (Fill and Reischl 2011). Potential future domains that seem particularly suited for such an approach are technology-intensive applications, e. g. in the context of Internet-of-Things and Industry 4.0, as well as for applications for regulatory and compliance-related matters. Although many applications so far have built on business process models, semantic annotations can be applied to any type of modeling language. In this context, especially modeling languages in the areas of enterprise architecture management, technical systems design, or knowledge management benefit from the loose coupling with additional information structures via semantic annotations.

Despite these advantages, there are some limitations of the approach. Although the handling of semantic annotations is easy to accomplish using the shown SeMFIS platform, some parts of the necessary technical infrastructure still have to be understood for their successful usage. This concerns currently the interaction with OWL ontologies, which are admittedly powerful but also difficult in their specification. As had already been remarked, the underlying logic needs to be thoroughly understood if this kind of ontologies shall be used to their full extent. These technical details could be further abstracted for the user. Or, only a sub-set of the full OWL language could be provided. Furthermore, in its current conception, the SeMFIS platform does not provide any

automation or recommendation facilities for suggesting ontology concepts to a user. This may be provided with third-party plugins but should be more tightly integrated into the overall annotation approach to make it easier for a user to choose from suitable annotations.

In addition to these interaction issues, the approach of semantic annotations as it has been discussed in this paper does not address the level of the modeling language directly. Although a way has been presented to circumvent this using the *All class instances* option, it may be preferable for certain applications to address the modeling language with semantic annotations in the same way. For example, this would permit to annotate attributes of the modeling language, thus allowing for a much more fine-grained extension of a modeling language. Another aspect concerns the current limitation to ontologies of only three types. With the upcoming of linked data and the accompanying standards, it may be favorable in the future to use in addition RDF and RDFS data sources for the annotation.

7 Conclusion and Outlook

In conclusion, we have presented in this paper an innovative approach for evolving modeling methods based on the concept of semantic annotations. For illustrating its feasibility, the approach has been technically realized on the ADOxx-based SeMFIS platform and applied to a use case in the area of risk management. Future work will include the application of the approach to further domains and modeling methods. It will be investigated how the approach could contribute to recently upcoming proposals for multi-level modeling in the area of enterprise modeling (Atkinson et al. 2013; Frank 2014b) as well as to novel forms of storing and processing model information using blockchain technologies (Fill and Härer 2018).

References

Atkinson C., Gerbig R., Fritzsche M. (2013) Modeling Language Extension in the Enterprise Systems Domain. In: Gašević D., Hatala M., Nezhad H., Reichert M. (eds.) International Enterprise Distributed Object Computing Conference. IEEE, pp. 49–58

Atkinson C., Gerbig R., Kuehne T. (2014) Comparing Multi-Level Modeling Approaches. In: Atkinson C., Grossmann G., Kühne T., De Lara J. (eds.) Multi-Level Modeling Workshop. CEUR Workshop Proceedings, pp. 53–62

Berners-Lee T., Hendler J., Lassila O. (2001) The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In: Scientific American 284(5), pp. 34–43

Blair G., Bencomo N., France R. (2009) Models@run.time. In: IEEE Computer 42(10), pp. 22–27

Bork D., Fill H.-G. (2014) Formal Aspects of Enterprise Modeling Methods: A Comparison Framework. In: Sprague R. H. (ed.) 47th Hawaii International Conference on System Sciences. IEEE, pp. 3400–3409

Born M., Doerr F., Weber I. (2007) User-Friendly Semantic Annotation in Business Process Modeling. In: Weske M., Hacid M., Godart C. (eds.) Web Information Systems Engineering – WISE 2007 Workshops Vol. 4832 LNCS. Springer, pp. 260–271

Braun R. (2015) Towards the state of the art of extending enterprise modeling languages. In: Desfray P., Filipe J., Hammoudi S., Pires L. F. (eds.) International Conference on Model-Driven Engineering and Software Development. IEEE, pp. 1–9

Bromiley P., McShane M., Nair A., Rustambekov E. (2015) Enterprise Risk Management: Review, Critique, and Research Directions. In: Long Range Planning 48(4), pp. 265–276

Chebotko A., Deng Y., Lu S., Fotohui F., Aristar A. (2007) An Ontology-based Multimedia Annotator for the Semantic Web of Language Engineering In: Semantic Web-Based Information Systems: State-of-the-Art Applications Idea Group Inc, pp. 140–160

Chen K., Sztipanovits J., Neema S. (2005) Toward a Semantic Anchoring Infrastructure for Domain Specific Modeling Languages. In: Proceedings of the 5th ACM International Conference on Embedded Software. ACM, pp. 35–43

Clark T., Kulkarni V., Barn B., France R., Frank U., Turk D. (2014) Towards the Model Driven Organization. In: Sprague R. H. (ed.) 47th Hawaii International Conference on System Science. IEEE, pp. 4817–4826

De Lara J., Guerra E. (2010) Generic Meta-Modelling with Concepts, Templates and Mixin Layers. In: Petriu D. C., Rouquette N., Haugen Ø. (eds.) 13th International Conference on Model Driven Engineering Languages and Systems. Springer, pp. 16–30

De Lara J., Guerra E., Cuadrado J. (2015) A-posteriori Typing for Model-Driven Engineering. In: Lethbridge T., Cabot J., Egyed A. (eds.) 18th International Conference on Model Driven Engineering Languages and Systems. IEEE, pp. 156–165

Del Fabro M. D., Bezivin J., Jouault F., Valduriez P. (2005) Applying Generic Model Management to Data Mapping. In: Journées Bases de Données Avancées (BDA), pp. 343–355

Demuth A., Lopez-Herrejon R., Egyed A. (2013) Supporting the Co-evolution of Metamodels and Constraints through Incremental Constraint Management. In: Moreira A., Schätz B., Gray J., Vallecillo A., Clarke P. (eds.) Model-Driven Engineering Languages and Systems. Springer, pp. 287–303

Desel J. (1997) How distributed algorithms play the token game In: Foundations of Computer Science, pp. 297–306

Di Battista G., Eades P., Tamassia R., Tollis I. (1999) Graph Drawing - Algorithms for the visualization of graphs. Prentice Hall, London

Engels G., Hausmann J., Heckel R., Sauer S. (2000) Dynamic Meta-Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML. In: Evans A., Kent S., Selic B. (eds.) UML 2000 — The Unified Modeling Language. Springer, pp. 323–337

Faisst U., Buhl H. (2005) Integrated Enterprise Balancing mit integrierten Ertrags- und Risikodatenbanken. In: Wirtschaftsinformatik 47(6), pp. 403–412

Feldmann S., Rösch S., Legat C., Vogel-Heuser B. (2014) Keeping requirements and test cases consistent: Towards an ontology-based approach. In: IEEE Conference on Industrial Informatics (INDIN). IEEE

Fill H.-G. (2009a) Design of Semantic Information Systems using a Model-based Approach. In: AAAI Spring Symposium - Social Semantic Web: Where Web 2.0 Meets Web 3.0. Technical Report SS-09-08. AAAI, pp. 19–24

Fill H.-G. (2009b) Visualisation for Semantic Information Systems. Gabler

Fill H.-G. (2011a) On the Conceptualization of a Modeling Language for Semantic Model Annotations. In: Salinesi C., Pastor O. (eds.) Advanced Information Systems Engineering Workshops Vol. 83 LNBIP. Springer, pp. 134–148

Fill H.-G. (2011b) Using Semantically Annotated Models for Supporting Business Process Benchmarking. In: Grabis J., Kirikova M. (eds.) 10th International Conference on Perspectives in Business Informatics Research Vol. 90 LNBIP. Springer, pp. 29–43

Fill H.-G. (2012a) An Approach for Analyzing the Effects of Risks on Business Processes Using Semantic Annotations. In: European Conference on Information Systems 2012. AIS

Fill H.-G. (2012b) Using Obfuscating Transformations for Supporting the Sharing and Analysis of Conceptual Models. In: Robra-Bissantz S., Mattfeld D. (eds.) Tagungsband der Multikonferenz Wirtschaftsinformatik 2012. GITO, pp. 1599–1612

Fill H.-G. (2017) SeMFIS: A Flexible Engineering Platform for Semantic Annotations of Conceptual Models. In: Semantic Web 8(5), pp. 747–763

Fill H.-G., Gericke A., Karagiannis D., Winter R. (2007) Modellierung für Integrated Enterprise Balancing. In: Wirtschaftsinformatik 06/2007, pp. 419–429

Fill H.-G., Härer F. (2018) Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling. In: Bui T. (ed.) HICSS'51. AIS, pp. 4045–4054

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

Fill H.-G., Redmond T., Karagiannis D. (2012) FDMM: A Formalism for Describing ADOxx Meta Models and Models. In: Maciaszek L., Cuzzocrea A., Cordeiro J. (eds.) Proceedings of International Conference on Enterprise Information Systems Vol. 3, pp. 133–144

Fill H.-G., Reischl I. (2011) Stepwise Semantic Enrichment in Health-related Public Management by Using Semantic Information Models. In: Smolnik S., Teuteberg F., Thomas O. (eds.) Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications Vol. 195–212. IGI Press

Fill H.-G., Schremser D., Karagiannis D. (2013) A Generic Approach for the Semantic Annotation of Conceptual Models using a Service-oriented Architecture. In: International Journal of Knowledge Management 9(1), pp. 76–88

Fitzgerald M., Kruschwitz N., Bonnet D., Welch M. (2013) Embracing Digital Technology - A New Strategic Imperative. In: MIT Sloan Management Review 55(2), pp. 1–12

Fox M., Grüninger M. (1998) Enterprise Modeling. In: AI Magazine 19(3), pp. 109–121

France R., Rumpe B. (2007) Model-driven Development of Complex Software: A Research Roadmap. In: Briand L. C., Wolf A. L. (eds.) Future of Software Engineering Conference. IEEE, pp. 37–54

Frank U. (2011) Some Guidelines for the Conception of Domain-Specific Modelling Languages. In: Nüttgens M., Thomas O., Weber B. (eds.) Enterprise Modelling and Information Systems Architectures: Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures Vol. P-190. GI, pp. 93–106

Frank U. (2014a) Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. In: Software and Systems Modeling 13(3), pp. 941–962

Frank U. (2014b) Multilevel Modeling – Toward a New Paradigm of Conceptual Modeling and Information Systems Design. In: Business & Information Systems Engineering 6(6), pp. 319–337

Fraser M., Kumar K., Vaishnavi V. (1994) Strategies for incorporating formal specifications in software development. In: Communications of the ACM 37(10), pp. 74–86

Fu B., Noy N., Storey M.-A. (2017) Eye tracking the user experience – An evaluation of ontology visualization techniques. In: Semantic Web 8(1), pp. 23–41

Gennari J., Musen M. A., Fergerson R., Grosso W., Crubezy M., Eriksson H., Noy N., Tu S. (2003) The evolution of Protégé: an environment for knowledge-based systems development. In: International Journal of Human-Computer Studies 58(1), pp. 89–123

Gericke A., Fill H.-G., Karagiannis D., Winter R. (2009) Situational Method Engineering for Governance, Risk and Compliance Information Systems. In: Vaishanvi V. (ed.) Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology. ACM

Gogolla M. (2015) Experimenting with Multi-Level Models in a Two-Level Modeling Tool. In: Atkinson C., Grossmann G., Kühne T., De Lara J. (eds.) Multi-Level Modelling Workshop. CEUR, pp. 3–12

Groth L. (1999) Future Organizational Design - The Scope for the IT-based Enterprise. John Wiley and Sons Ltd.

Harel D., Rumpe B. (Aug. 2000) Modeling Languages: Syntax, Semantics and All That Stuff - Part I: The Basic Stuff. MCS00-16. The Weizmann Institute of Science

Harel D., Rumpe B. (2004) Meaningful Modeling: What's the Semantics of "Semantics"? In: IEEE Computer October 2004, pp. 64–72

Henderson-Sellers B. (2012) On the mathematics of modelling, metamodelling, ontologies and modelling languages. Springer

Herbst J., Junginger S., Kühn H. (1997) Simulation in Financial Services with the Business Process Management System ADONIS. In: Proceedings of the 9th European Simulation Symposium. Society for Computer Simulation, pp. 491–495

Horrocks I., Patel-Schneider P., Van Harmelen F. (2003) From SHIQ and RDF to OWL: The Making of a Web Ontology Language. In: Web Semantics: Science, Services and Agents on the World Wide Web 1(1), pp. 7–26

Hui G. (2014) How the Internet of Things Changes Business Models. In: HBR.org July 29

Jahn M. (2014) Evolution von Meta-Modellen mit sprachbasierten Mustern. PhD thesis, University of Bayreuth

Karagiannis D., Fill H.-G., Höfferer P., Nemetz M. (2008) Metamodeling: Some Application Areas in Information Systems. In: Kaschek R., Kop C., Steinberger C., Fliedl G. (eds.) Information Systems and e-Business Technologies. Springer, pp. 175–188

Karagiannis D., Kühn H. (2002) Metamodeling Platforms In: Third International Conference EC-Web 2002 – Dexa 2002 Bauknecht K., Min Tjoa A., Quirchmayr G. (eds.) Springer, p. 182

Khelladi D., Hebig R., Bendraou R., Robin J., Gervais M.-P. (2015) Detecting Complex Changes During Metamodel Evolution. In: Zdravkovic J., Kirikova M., Johannesson P. (eds.) International Conference on Advanced Information Systems Engineering 2015. Springer, pp. 263–278

Kolovos D., Rose L., Matragkas N., Paige R., Polack F., Fernandes K. (2010) Constructing and Navigating Non-invasive Model Decorations. In: Tratt L., Gogolla M. (eds.) Theory and Practice of Model Transformations. Springer, pp. 138–152

Krahn H., Rumpe B. (2006) Towards Enabling Architectural Refactorings through Source Code Annotations. In: Mayr H. C., Breu R. (eds.) Modellierung 2006 Vol. 82. GI-LNI, pp. 203–212

Langer P., Wieland K., Wimmer M., Cabot J. (2012) EMF Profiles: A Lightweight Extension Approach for EMF Models. In: Journal of Object Technology 11(1), pp. 1–29

Lin Y. (2008) Semantic Annotation for Process Models: Facilitating Process Knowledge Management via Semantic Interoperability. PhD thesis, Norwegian University of Science and Technology (NTNU)

Moody D. (2009) The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In: IEEE Transactions on Software Engineering 35(6), pp. 756–779

Musen M. A. (2015) The protégé project: a look back and a look forward. In: AI Matters 1(4), pp. 4–12

Obrst L. (2003) Ontologies for semantically interoperable systems. In: International Conference on Information and Knowledge Management. ACM, pp. 366–369

OMG (2011) Business Process Model and Notation (BPMN) Version 2.0.. Object Management Group. Last Access: http://www.omg.org/spec/BPMN/2.0/PDF/ 01-03-2011

OMG (2015a) Unified Modeling Language Version 2.5.. Object Management Group. Last Access: http://www.omg.org/spec/UML/2.5/PDF 23-06-2016

OMG (2015b) XML Metadata Interchange (XMI) Specification Version 2.5.1.. Object Management Group. Last Access: http://www.omg.org/spec/XMI/2.5.1/PDF/ 21-06-2016

Porter M., Heppelmann J. (2015) How Smart, Connected Products Are Transforming Companies. In: Harvard Business Review (October 2015), pp. 96–112

Raineri A. (2011) Change management practices: Impact on perceived change results. In: Journal of Business Research 64, pp. 266–272

Rigby D., Bilodeau B. (2015) Management Tools and Trends 2015.. Bain & Company. Last Access: http://www.bain.com/publications/articles/management-tools-and-trends-2015.aspx last accessed 08-06-2016

Rosemann M. (2006) Potential pitfalls of process modeling: part B. In: Business Process Management Journal 12(3), pp. 377–384

Rossi M., Ramesh B., Lyytinen K., Tolvanen J.-P. (2004) Managing Evolutionary Method Engineering by Method Rationale. In: Journal of the AIS 5(9), pp. 356–391

Sandkuhl K., Fill H.-G., Hoppenbrouwers S., Krogstie J., Matthes F., Opdahl A., Schwabe G., Uludag Ö., Winter R. (2018) From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. In: Business & Information Systems Engineering 60(1), pp. 69–80

Sandkuhl K., Stirna J., Persson A., Wissotzki M. (2014) Enterprise Modeling - Tackling Business Challenges with the 4EM Method. Springer

Smaragdakis Y., Batory D. (2002) Mixin Layers: An Object-Oriented Implementation Technique for Refinements and Collaboration-Based Designs. In: ACM Transactions on Software Engineering and Methodology 11(2), pp. 215–255

Strecker S., Heise D., Frank U. (2011) RiskM: A multi-perspective modeling method for IT risk assessment. In: Information Systems Frontiers 13(4), pp. 595–611

Suriadi S., Weiss B., Winkelmann A., ter Hofstede A., Adams M., Conforti R., Fidge C., La Rosa M., Ouyang C., Rosemann M., Pika A., Wynn M. (2014) Current research in risk-aware business process management : overview, comparison, and gap analysis. In: Communications of the Association for Information Systems 34(1), pp. 933–984

Tarski A. (1936) Der Wahrheitsbegriff in den formalisierten Sprachen In: Studia Philosophica I, pp. 261–405

Tjoa S., Jakoubi S., Goluch G., Kitzler G., Goluch S., Quirchmayer G. (2011) A Formal Approach Enabling Risk-Aware Business Process Modeling and Simulation. In: IEEE Transactions on Services Computing 4(2), pp. 153–166

Trojer T., Farwick M., Haeusler M. (2014) Modeling Techniques for Enterprise Architecture Documentation: Experiences from Practice. In: Atkinson C., Grossmann G., Kühne T., De Lara J. (eds.) Multi-Level Modeling Workshop, pp. 113–128

W3C (2012) OWL 2 Web Ontology Language Document Overview (Second Edition) - W3C Recommendation 11 December 2012.. World Wide Web Consortium (W3). Last Access: https://www.w3.org/TR/owl2-overview/ 01-02-2015

Wachsmuth G. (2007) Metamodel Adaptation and Model Co-adaptation. In: Ernst E. (ed.) ECOOP 2007 – Object-Oriented Programming. Springer, pp. 600–624

Weiss B., Winkelmann A. (2011) Developing a Process-Oriented Notation for Modeling Operational Risks - A Conceptual Metamodel Approach to Operational Risk Management in Knowledge Intensive Business Processes within the Financial Industry. In: Sprague R. (ed.) Hawaii International Conference on System Sciences. IEEE, pp. 1–10

Wile D. (2001) Supporting the DSL Spectrum. In: Journal of Computing and Information Technology 9, pp. 263–287

Zur Muehlen M., Rosemann M. (2005) Integrating Risks in Business Process Models. In: 16th Australasian Conference on Information Systems