

Transforming Enterprise Models to Linked Data via Semantic Annotations

Benedikt Pittl¹, Hans-Georg Fill²

Abstract: The use of conceptual models in enterprises is today a well-known fact. This includes many different types of models ranging from process models, organizational models, and infrastructure models to various types used in software engineering and technical systems development. Although these models are largely specified in a formal or at least semi-formal way, the knowledge contained in them is often only accessible via manual inspection. The primary reason for this shortcoming is the use of different formats for expressing models and the lack of machine-processable semantic specifications of the model content. In this paper we present a flexible approach for transforming information from such enterprise models to RDF. Thereby, we use a model weaving technique to annotate conceptual models with concepts from ontologies. For assessing its technical feasibility, the approach has been prototypically implemented on the SeMFIS platform and applied to a use case in the area of business process management.

Keywords: Conceptual Model; RDF; Ontology; Semantic Annotation

1 Introduction

Today, enterprises heavily rely on conceptual models such as business process models, organization models or infrastructure models, thus potentially leading to hundreds if not thousands of models just within one organization [Ro06, WH01]. Such models are often created with the aim of fostering communication and understanding [My92] and are an important source of knowledge. Usually, these models are stored in the databases of the used modeling tools [vDDM13]. For analyzing these models using query techniques and benchmarks [APW08, EKO07, vDDM13], or for executing them [Fi12], models need to be available in a machine-processable format that is ideally based on a standard representation. In the context of the Web of Data, conceptual models were recently identified as a valuable source for data repositories [BK16]. Thereby, the model content is transformed to ontologies - usually in RDF format. The transformation to ontologies has two main benefits: (i) *Exchange of Model Information*. Standardized formats such as RDF foster the exchange of models across different tools and platforms. (ii) *Semantic Processing*. The usage of ontology formats

¹ University of Vienna, Faculty of Computer Science, Waehringerstrasse 29, 1090 Vienna, Austria; benedikt.pittl@univie.ac.at

² University of Bamberg, Department for Information Systems - System Development and Database Application Group, An der Weberei 5, 96047 Bamberg, Germany; hans-georg.fill@uni-bamberg.de

enables semantic processing based on reasoning and query techniques, especially through linking the model information to other ontologies and linked data repositories. This enables the linkage, querying and merging of different data sources with information contained in the models [BHB09].

The scientific community proposed two main paradigms for the transformation from conceptual models to ontologies used in the Web of Data: (i) Hinkelmann [Hi15] presented seven approaches which describe how to establish a linkage between conceptual models and ontologies. These approaches range from *simple links* – through adding textual attributes in model elements which contain a URI to an ontology element – over *semantic tunnels* where semantic information for model elements is retrieved via webservice – to *semantic transit models* where models contain references to an ontology. However, a concrete approach for transforming models to RDF was not described. Additionally, all seven approaches have the drawback that a new modeling language is required that permits to link elements to ontology concepts. Based on the assumption that enterprises already have large model repositories [WH01], the requirement of such a new modeling language would lead to a considerable effort for remodeling or at least migrating the existing models. (ii) For overcoming these drawbacks, an *RDFizer* has been presented for transforming conceptual models to RDF [BK16]. Thereby, static transformation patterns were defined that are applicable to arbitrary models. While this approach is comprehensive in the sense that each model element and attribute is serialized according to the pattern, it is not easily adaptable to specific needs e.g. to define how model concepts are serialized to RDF. Furthermore, the modeling languages of the models to be transformed has to be altered if additional RDF information is to be represented.

The research question which we want to answer in this paper is "*How to semantically enrich and process existing visual models in standardized semantic formats?*". Thereby, we pursue the following three goals: (i) The approach has to be generic so that it is applicable for models created with different modeling languages. (ii) The approach has to be simple so that business users are able to do the enrichment (iii) The approach has to be adaptive so that modifications of the enrichment are possible. Hence, in this paper we present a *customized model weaving approach* for transforming the content of conceptual models to RDF. Our approach does not require an adaptation of existing modeling languages but allows referring to existing ontology concepts. Thereby, our weaving approach is based on semantic annotations for linking model elements and their attributes with ontology schema concepts. The annotations can be created, removed or modified without affecting the conceptual models nor the ontology, which makes our approach useful for semantically enriching and processing *already existing models*. For easing the specification of the semantic annotations, we provide a domain-specific visual language. We conceptualized and evaluated the approach following three steps: (i) Specification of a visual language for configuring flexible transformations via annotations (ii) Specification of generic rules for conducting the transformation to RDF (iii) Implementation of the approach using the SeMFIS platform [Fi17].

The remainder of the paper is structured as follows. An overview of existing paradigms for semantically enriching models is presented in section 2. Our transformation approach based on visual annotations is explained in section 3. Section 4 describes the technical implementation followed by a use case based evaluation in section 5. A discussion is described in section 6. The paper closes with a conclusion in section 7.

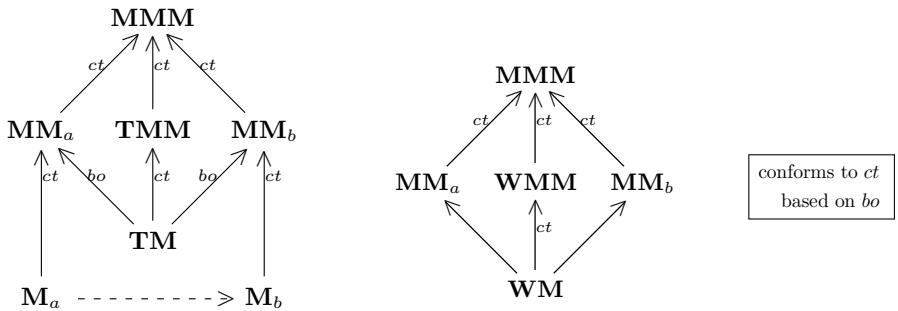
2 Background and Related Work

In the literature there are currently two research directions being investigated for linking ontologies with conceptual models: *model transformation* and *model weaving* [Fi11]. We will thus first describe the differences between model transformation and model weaving. Second, we will review existing weaving approaches which define how models can be semantically enriched. In the third part we will investigate existing approaches for the model-to-ontology transformation with a special focus on RDF ontologies.

The transformation between different types of models has been discussed to a large extent in the context of Model Driven Engineering [DFBV06]. Thereby, transformation models are models which describe operations for transferring source models to target models. These operations are executed by a transformation engine. Prominent examples of this approach are transformations via Query View Transformation (QVT) or the ATL Transformation Language. An overview of the model transformation approach from a generic perspective is shown in Figure 1a. The source model (\mathbf{M}_a) and the target model (\mathbf{M}_b) conform to the metamodels \mathbf{MM}_a and \mathbf{MM}_b which is illustrated with the *ct* connectors. Similarly, the transformation models (\mathbf{TM}) conform to a metamodel (\mathbf{TMM}). All three metamodels conform to a meta-metamodel. The transformation operations are part of the transformation model which references elements of the metamodels \mathbf{MM}_a and \mathbf{MM}_b . This is illustrated with the *based-on* (*bo*) connector. For example, the ATL rules used for a model transformation are grouped to a transformation model \mathbf{TM} which is executed by an ATL transformation engine.

The second research direction for linking models and ontologies is to use weaving models. The main difference between model weaving and model transformation is that transformation metamodels have *fixed semantics* that can be implemented by transformation engines, whereas weaving models have *user-defined semantics* [DFBV06]. The model weaving approach is illustrated in Figure 1b. For model weaving, three metamodels are used whereby \mathbf{WMM} is the weaving metamodel. Weaving models (\mathbf{WM}) are models which use domain specific link types for establishing references between two metamodels (\mathbf{MM}_a and \mathbf{MM}_b). The overall goal of model weaving is just to establish links between elements of two models. The weaving model can be used for model transformations but it is not limited to it. Hence, the illustration in Figure 1b does not show a transformation example as Figure 1a. Weaving approaches are e.g. used for model traceability as well as for model alignment. According to [DFBV06] model weaving fulfills the following requirements: (i) the weaving model supports the expression of links between two model elements, (ii) different types of links

have to be supported whereby the link type provides the semantics, (iii) the links support different arities, (iv) and the links have references to the model elements.



(a) Model transformation based on [Jo06] (b) Model weaving from [DFJ05]

Fig. 1: Model transformation and model weaving

Based on these foundations we can now investigate approaches that make use of these concepts for linking conceptual models and ontologies. Two main reasons can be stated why such connections are beneficial: First, the use of standardized exchange formats such as RDF and OWL permits the easy transfer of model information across different tools and platforms. Second, ontology formats permit semantic processing based on reasoning and query techniques, especially through linking the model information to other ontologies and linked data repositories.

Following the direction of model transformations, it is often being referred to the XML Metadata Interchange (XMI) format as a starting point for transforming models to ontologies. XMI is a standardized format maintained by the OMG³ which fosters the exchange of models between different modeling tools. For example, [Ga04] describe a transformation approach for models represented in XMI to OWL via XSLT. Similar approaches are described in [BB12] and [Cr01]. In [TF07] Event driven Process Chains (EPC) models are transformed to an RDF ontology. Thereby, the authors assume that the EPC model is stored in the XML-based format EPML so that the transformation to RDF-XML can also revert to XSLT. However, all these XSLT transformations are static. They are predefined and cannot be adapted by end users. Furthermore, such direct XSLT transformations consider metamodels only implicitly. Linkages to other ontologies or linked data repositories are not foreseen.

In [BK16] a transformation approach from models to RDF is introduced, denoted as *RDFizer*. It supports three different ways for first linking existing URIs to model elements: (i) *Linking by Equivalence*: This way is very similar to the direct linkage approach presented in [Hi15]. It expects a string attribute in each model element which contains a URI of an equivalent ontology concept. (ii) *Linking by Modeling Properties*: Similar to the previous approach,

³ <http://www.omg.org/spec/XMI/>

URIs are entered into string attributes of model elements. However, the interpretation is different. The entered URI does not represent the model element which contains the attribute. Instead, it refers to related concepts. (iii) *Linking by Arbitrary Properties or Types*: For adding additional information to model elements, an attribute Table can be added. This Table is used for generating customized RDF triples. Thereby, the model element which contains the Table is either the subject or the object of the triple. In a second step, [BK16] then apply static patterns for the transformation of the conceptual models to RDF ontologies. These patterns are pre-defined generic rules which determine how model information is transformed to RDF triples. Using this approach, customization is possible ex-post, e.g. using the external Java component "RDF export customizer" as shown in [KB16] and [BK15]. It allows adding, removing or modifying RDF triples created with the patterns. In addition to the RDFizer several related approaches exist. For example, in [Ka06] an approach for semantic lifting of metamodels was introduced. Thereby, the authors transfer metamodels to ontologies using mapping patterns. Based on this mapping metamodels are transferred to ontologies.

In the following we investigate approaches which make use of the model weaving paradigm. The previously mentioned seven different approaches by Hinkelmann [Hi15] do not fulfill all described requirements for model weaving as stated above - a detailed discussion of them is out of the scope of this paper. However, they are closely related to this direction. In the following, we focus on the three most relevant types. In all of them it is implied that model elements contain references to ontology concepts: (i) *Indirect Linkage*: Following this approach, the whole ontology is represented as a visual model called *semantic transit model*. The connections between the model and the ontology are established using additional hyperlink attributes in the model. (ii) *Direct and Indirect Linkage*: This is a combination of the indirect linkage approach and the so-called *semantic tunnel approach*. It retrieves ontology concepts via a webservice (semantic tunnel). Thereby, only selected concepts are represented in the semantic transit model. The model elements have again hyperlink attributes for referencing ontology concepts. Based on these references additional ontology concepts can be retrieved and offered to the user. (iii) *Loose Coupling*: This approach introduces an intermediate ontology that acts as a reference for connecting it to model elements.

In summary it can be stated that existing approaches have achieved various ways for transforming conceptual models to ontology formats. However, an approach for transforming conceptual models to RDF with a special focus on (i) adaptability (ii) semantic enrichment for linking them to existing ontology and data repositories and (iii) adequacy for non-technical users is missing so far.

3 Transformation via Visual Annotations

The main motivation for our work is based on the assumption that potentially large repositories of conceptual models already exist in an organization which are used by

human actors and technical systems alike, cf. [WH01]. These models are usually stored in vendor-specific formats and there is a lack of machine-processable semantic specifications of the model content. Rather than remodelling all existing models with an adequate modeling language - which would be costly - we could semantically enrich the existing models. Therefore, we first need to provide means for a semantic enrichment of these models to align them with existing data and ontology schemas. Semantic annotations have been shown as a solution that does not require changes in the models nor the underlying modeling language [Fi11]. As depicted in Figure 2, we use annotations that are stored in visual Annotation Models on the *Configuration Layer* of our approach. Thereby, they configure the RDF serialization of conceptual models. The annotations have references to both, model concepts and ontology schema concepts. The RDF serialization itself belongs to the second layer called *Standardized Semantic Representation Layer*. The resulting RDF can subsequently be merged with other ontologies or queried, which is foreseen in the *Analysis Layer*.

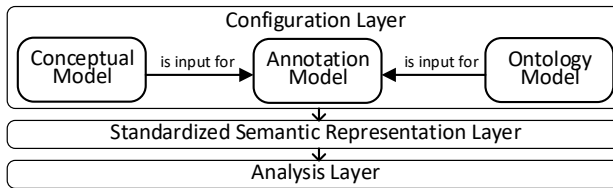


Fig. 2: Overview of the Three-layered Approach for Transforming Conceptual Models to RDF

As the annotations are created as visual models, the provision of a corresponding visual modeling language is required. For describing the visual modeling language as well as the transformations to RDF in a non-ambiguous way we will revert to the FDMM formalism [FRK12] in the following. A summary of the notation and the elements of the visual language - which are described using FDMM - is given in Table 1.

3.1 FDMM Core Concepts

For formally describing modeling languages, different formalisms have been introduced – e.g. for EMF [Sc08] or in the OCL specification⁴. We decided to use the FDMM formalism as it aligns well with the concepts used in the ADOxx metamodeling platform, which we used later for the implementation and evaluation of our approach [FRK12].

FDMM describes metamodels \mathbf{MM} using four components $\mathbf{MM} = \langle \mathbf{MT}, \leq, \text{domain}, \text{range}, \text{card} \rangle$. Each metamodel consists of a set model types \mathbf{MT} which are used to create a set of model instances \mathbf{mt} . Each model type \mathbf{MT}_i consists of a set of object types \mathbf{O}_i^T , which have in turn a set of attributes \mathbf{A}_i . Each attribute is assigned a datatype

⁴ <http://www.omg.org/spec/OCL/2.0/>

from the set \mathbf{D}_i^T . So, in FDMM a model type is described as follows: $\mathbf{MT}_i = \langle \mathbf{O}_i^T, \mathbf{D}_i^T \mathbf{A}_i \rangle$. \leq is an ordering on the set of object types \mathbf{O}_i^T for defining an object type hierarchy similar to inheritance hierarchies in object-orientation. *domain* is a function which assigns attributes to object types. The *range* function assigns datatypes to attributes while the *card* function defines the cardinality of attribute values. Models \mathbf{mt}_i consist of triples τ representing the model content. The first element of a triple $t \in \tau$ represents an instance of an object type, the second component represents an attribute, and the last component represents the attribute value. Due to limited space we do not describe FDMM in more detail here - for more information we refer to [FRK12].

3.2 Formalizing Visual Annotations in FDMM

The visual modeling language for annotations consists of a single model type \mathbf{MT}_{Annot} . This model type has a set of object types \mathbf{O}_{Annot}^T which have a set of attributes \mathbf{A}_{Annot} which have in turn data types \mathbf{D}_{Annot}^T . The object types used in the model type are described in the following equation. In FDMM, model connectors such as *isInputFor* and *refersTo* are also considered as object types.

$$\mathbf{O}_{Annot}^T = \{ModelReference, ConnectorReference, AttributeReference, \\ OntologyReference, Annotator, AnnotationElement, \\ ModelReferences, isInputFor, refersTo\} \quad (1)$$

All the attributes used in the object types are part of the set \mathbf{A}_{Annot} .

$$\mathbf{A}_{Annot} = \{Name, AllClassInstances, InstanceReference, AttributeName, \\ ConnectorName, AnnotationType, isInputFor-from, \\ isInputFor-to, OntologySchemaConceptReference, \\ refersTo-from, refersTo-to\} \quad (2)$$

The datatypes of the attributes are summarized in the set \mathbf{D}_{Annot}^T . $Enum_{anntype}$ represents an enumeration list.

$$\mathbf{D}_{Annot}^T = \{String, Enum_{anntype} = \{instanceOf, isEqualTo, isBroaderThan, \\ isNarrowerThan, isSubclassOf, isSuperclassOf, \\ isInstanceUsingFromClass, isInstanceUsingToClass\}\} \quad (3)$$

We have defined an ordering of the object types similar to an inheritance hierarchy to avoid the duplicate specification of attributes:






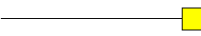

$$\begin{aligned} AnnotationElement &\leq InstanceType \\ OntologyReference &\leq AnnotationElement \\ Annotator &\leq AnnotationElement \\ ModelReferences &\leq AnnotationElement \\ ConnectorReference &\leq ModelReferences \\ ModelReference &\leq ModelReferences \\ AttributeReference &\leq ModelReference \end{aligned} \quad (4)$$

The object type *InstanceType* can be considered as a super-object type similar to the class *Object* in Java - so we did not list it explicitly in \mathbf{O}_{Annot}^T . Attributes and their value ranges were specified using the FDM domain and range functions [FRK12]: $domain(Name) = \{AnnotationElement\}$, $domain(AllClassInstances) = \{ModelReferences\}$, $domain(InstanceReference) = \{ModelReferences\}$, $domain(AttributeName) = \{AttributeReference\}$, $domain(AnnotationType) = \{Annotation\}$, $domain(OntologySchemaConceptReference) = \{OntologyReference\}$, $domain(isInputFor-from) = \{isInputFor\}$, $domain(isInputFor-to) = \{isInputFor\}$, $domain(refersTo-from) = \{refersTo\}$, $domain(refersTo-to) = \{refersTo\}$. $range(Name) = \{String\}$, $range(AllClassInstances) = \{true, false\}$, $range(InstanceReference) = \{InstanceType\}$, $range(AttributeName) = \{Enum_{attribute\ names}\}$, $range(AnnotationType) = \{Enum_{annotation\ type}\}$, $range(OntologySchemaConceptReference) = \{InstanceType\}$, $range(isInputFor-from) = \{ModelReferences\}$, $range(isInputFor-to) = \{Annotator\}$, $range(refersTo-from) = \{Annotator\}$, $range(refersTo-to) = \{OntologyReference\}$. In FDM the cardinality function - abbreviated with *card* - defines how many attribute values an object type can have. In our modeling type all attributes have at most one value. Therefore, we do not list the cardinality functions explicitly here.

3.3 Graphical Notation

The graphical notation of the described object types is depicted in Table 1. There are three different object types which have hyperlinks to classes, connectors and attributes of conceptual models: The Model Reference (MREF) object type has a hyperlink to model elements (instances of object types except connectors). The Attribute Reference (AREF) object type has a hyperlink to attributes of model elements and the Connector Reference (CREF) object type has a hyperlink to instances of connectors, which are object types in FDM. All three object types have further attributes for a more precise description of the type of linkage which should be established. For example, the MREF, AREF as well as the CREF object type have an attribute *applies to all instances*. This attribute indicates if the reference is only representative for the model element to which the MREF, AREF or CREF instances points to, or if it is representative for all instances of the same object type in the conceptual model. Instances of the object type *Annotator* are the connecting link between CREF, AREF and MREF elements which reference to contents of conceptual models and elements which reference to ontology schema concepts. The latter are instances of the OREF object type in our visual language. Annotator elements contain additional information regarding the type of linkage which is established. The connectors shown on the lower right corner of Table 1 are the connectors for constructing annotations. An example of an annotation created with our visual language is shown in the use case (section 5).

Similar to the transit model approach described in [Hi15] we make use of visual ontology models. This means that we represent ontologies such as OWL ontologies or frames ontologies as visual models. To ensure interoperability with applications such as Stanford

Graphical Notation	Description	Graphical Notation	Description
 <p>Modeltype: Business process model Model: Business process model -... Class: Activity Reference applies to all instances.</p>	<p>MREF- references to instances of model object types (non-connectors) FDMM: $ModelReference \in O^T_{Annot}$</p>	 <p>Modeltype: Business process model Model: Business process model -... Class: Activity Reference applies to all instances. Attribute: Description</p>	<p>AREF- references to attributes of instances of object types FDMM: $AttributeReference \in O^T_{Annot}$</p>
 <p>Modeltype: Business process model Model: Business process model -... Relation Class: Subsequent Reference applies to all instances.</p>	<p>CREF- references to instances of model object types (connectors) FDMM: $ConnectorReference \in O^T_{Annot}$</p>	 <p>Modeltype: Ontology Model Model: Ontology Model - new (2) Class: Class Instance: BPMNActivity</p>	<p>OREF- references to ontology schema concepts FDMM: $OntologyReference \in O^T_{Annot}$</p>
	<p>Annnotator- connecting MREF, AREF, CREF elements with OREF elements FDMM: $Annot \in O^T_{Annot}$</p>		<p>Connector between MREF, CREF or AREF and Annotator FDMM: $isInputFor \in O^T_{Annot}$</p>
			<p>Connector between Annotator and OREF FDMM: $refersTo \in O^T_{Annot}$</p>

Tab. 1: Overview of the Object Types Used in the Visual Language for Creating Annotations

Protégé we developed an import as well as an export function to standardized ontology serialization syntaxes (e.g. OWL-XML).

3.4 Transformation Rule in FDMM

After the annotations are created they are used in the standardized semantic representation layer to transform the annotated conceptual models to an RDF ontology. For a better understanding of how the transformation works we present in the following an exemplary transformation rule for annotations. It is assumed that an annotation model is present in which an MREF element is connected with an OREF element via an annotator element of type *instanceOf*. The MREF elements reference elements in the conceptual model and the OREF element corresponding ontology concepts.

The function *getAttributeValue* returns the attribute value t_3 (third component) of a triple $t \in \tau$ whereby $\mathbf{mt}_i \in \mu_{MT}(MT_{Annotation})$. We used FDMM also for describing OWL ontologies ($\mathbf{mt}_{OWL} \in \mu_{MT}(MT_{OWL})$) as well as the resulting RDF ontologies ($\mathbf{mt}_{RDF} \in \mu_{MT}(MT_{RDF})$). Similarly, we described the conceptual model $\mathbf{mt}_{REF} \in \mu_{MT}(MT_{REF})$ to which the MREF element refers to in FDMM. The Id attribute of the referenced elements

represents a unique identifier. A sample transformation rule is then specified as follows (the structure of the other rules is similar):

Transformation Rule to RDF described in FDMM: Annotation which connects MREF elements with OREF elements via an annotator of type instanceOf

$$\begin{aligned}
& \forall \mathbf{mt} \in \mu_{\mathbf{MT}}(MT_{\text{Annotation}}) \\
& \forall \mathbf{mref} \in \mu_{\mathbf{O}}(MREF, MT_{\text{Annotation}}) | (\mathbf{mref}, AllClassInstances, false) \in \beta(\mathbf{mt}) \\
& \forall \mathbf{schemaConcept} \in \mathbf{O} | (\\
& \quad \exists \mathit{inputConnector} | (\mathit{inputConnector}, \mathit{isInputFor-from}, \mathbf{mref}) \in \beta(\mathbf{mt}) \wedge \\
& \quad \exists \mathit{annotation} | (\mathit{inputConnector}, \mathit{isInputFor-to}, \mathit{annotation}) \in \beta(\mathbf{mt}) \wedge \\
& \quad \exists \mathit{refersToConnector} | (\mathit{refersToConnector}, \mathit{refersTo-from}, \mathit{annotation}) \in \beta(\mathbf{mt}) \wedge \\
& \quad \exists \mathit{oref} | (\mathit{refersToConnector}, \mathit{refersTo-to}, \mathit{oref}) \in \beta(\mathbf{mt}) \wedge \\
& \quad (\mathit{oref}, \mathit{OntologySchemaConceptReference}, \mathbf{schemaConcept}) \in \beta(\mathbf{mt}) \\
& \quad) \\
& \forall \mathbf{modelElement} \in \{ \mathit{getAttributeValue}(i) | i \in \{\beta(\mathbf{mt})\}_{t_1 = \mathbf{mref}} \\
& \quad \wedge t_2 = \mathit{InstanceReference} \} \\
& \quad \implies \\
& \quad \exists \mathbf{mt}_{\text{RDF}} \in \mu_{\mathbf{MT}}(MT_{\text{RDF}}) \wedge \\
& \quad \exists t \in \beta(\mathbf{mt}_{\text{RDF}}) | (\\
& \quad \quad t_1 \in \mu_{\mathbf{O}}(\mathit{Description}, MT_{\text{RDF}}) \wedge \\
& \quad \quad (t_1, \mathit{rdf} : \mathit{about}, y) \in \beta(\mathbf{mt}_{\text{RDF}}) | (\mathbf{modelElement}, \mathit{Id}, y) \in \beta(\mathbf{mt}_{\text{REF}}) \wedge \\
& \quad \quad (t_1, \mathit{rdf} : \mathit{type}, x) \in \beta(\mathbf{mt}_{\text{RDF}}) | (\mathbf{schemaConcept}, \mathit{Id}, x) \in \beta(\mathbf{mt}_{\text{OWL}}) \\
& \quad \quad) \\
& \quad)
\end{aligned}$$

In the same way we created FDMM-based rules for all kinds of annotations, i.e. with CREF elements for referencing connectors and AREF elements for referencing attributes. Due to the space limit these are omitted here.

4 Technical Implementation

Based on the FDMM specifications we prototypically implemented the approach using the SeMFIS platform [Fi17] to evaluate its technical feasibility. The reasons for using the ADOxx based SeMFIS platform are besides familiarity due to previous projects with this platform twofold: (i) ADOxx is open and (ii) ADOxx is widely used in the modeling community e.g. for the process modeling toolkit ADONIS. We extended SeMFIS with our RDF transformation approach. Additionally, we implemented an OWL/XML import/export function. The implementation follows a three-layer approach as shown in Figure 3. The model types were implemented using the ADOxx development toolkit underlying SeMFIS. The transformation rules were encoded using XSLT. In addition, a Java component was created for merging the OWL and RDF ontologies.

The following numbers correspond to the numbers used in Figure 3. (1,2) First, the conceptual models are created or loaded. Additionally, the visual ontology model is created from scratch or imported from an existing ontology file. (3) After the conceptual model as well as the ontology model are in place, annotations are created using the visual language

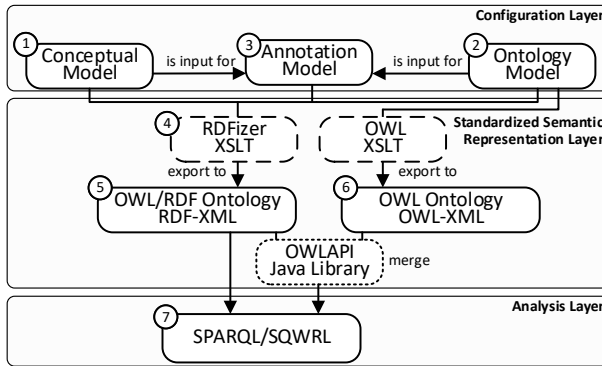


Fig. 3: Overview of the Technical Implementation

introduced in the previous section. Thereby, the content of conceptual models is linked with ontology schema constructs using annotations. (4) The annotations created with the visual language are used to configure the RDF serialization. Therefore transformation rules as exemplary introduced in the previous section are applied. (5) The transformation results in an RDF ontology containing the model elements for which we created annotations. The RDF ontology does not contain schema constructs but only instances. (6) It is thus possible to export the visual ontology model to which the annotations reference as an OWL ontology. (7) Then we get two ontologies: an RDF ontology (see (5)) and an OWL ontology (see (6)). Both ontologies can be merged. This is accomplished using the Java library OWLAPI. The resulting ontology then contains both, the OWL ontology including the schema constructs, as well as the instances stored as RDF triples. This ontology can then be processed using further semantic tools and techniques such as reasoners, query or rule engines.

5 Evaluation through a Use Case

In addition to the evaluation of the technical feasibility, we applied the approach to a use case to assess whether it can be used in a practical scenario.

For this purpose we reverted to an account opening business process that has been previously used within the Open Models Initiative⁵ - see [KMM16] for more information. Figure 4 shows an excerpt of the process model which was created with the domain-specific modeling language BPMS [KJS96]. Typically, models in such domain-specific languages are stored in an internal, platform-dependent serialization format, which makes machine-processing difficult. Therefore, the use of a standardized format is beneficial. Hence, we annotated the process model with OWL ontologies as shown in the right part in Figure 4. The depicted *OntoRule Ontology* ontology is an excerpt of the process ontology developed

⁵ <http://www.semfis-platform.org/>

within the OntoRules project⁶. We further created an ontology called *user ontology* with the data property *hasExecutionTime*. The lines shown in Figure 4 depict references as used in the MREF, CREF and OREF elements. To keep the Figure simple we have not shown the references of the AREF elements. As the annotation model shows, the process activity of the process model is annotated with the OWL class *Task*. Further, we used an AREF element for the annotation of the execution time attribute. It is annotated with the OWL dataproperty *hasExecutionTime*. The connector of the type *Subsequent*, which connects activities in a business process model, is annotated with the *follows* OWL property.

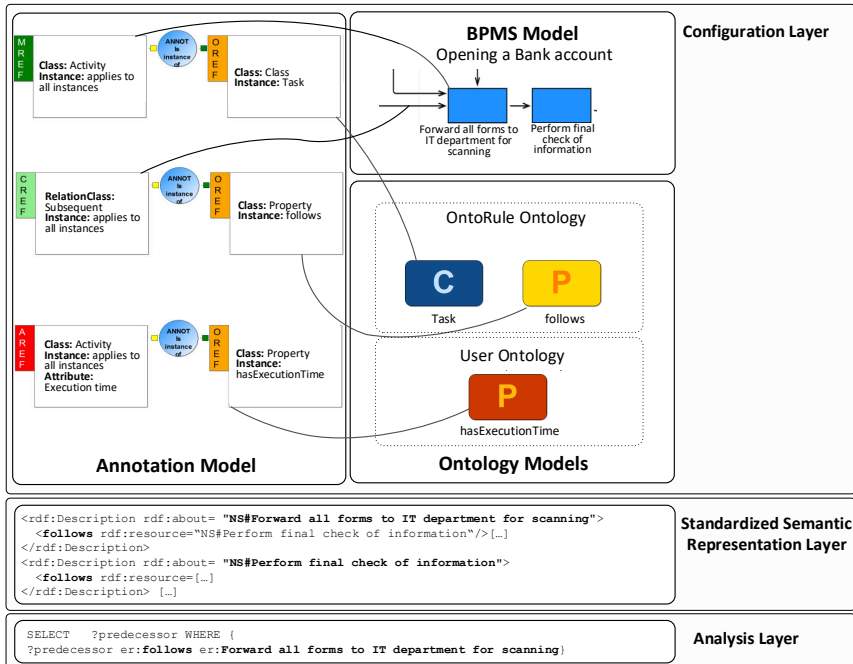


Fig. 4: Overview of Use Case Showing Excerpts of an Annotation Model Instance (left), of a Process Model in BPMS Notation (top right), Two Ontology Models (mid right), the Resulting RDF Representation, and a Sample Query in SPARQL

Based on these visual annotations, the XSLT stylesheets together with the Java component as described in the previous section transform the process model into an RDF serialization. This is depicted in the standardized semantic representing layer in Figure 4. Listing 1 shows a more detailed excerpt of the resulting RDF. For all annotated model elements, RDF resources are created including the corresponding types. The connector reference leads to the creation of the *follows* property. Similarly, the annotation of the attribute leads to the creation of the *hasExecutionTime* property. The serialized RDF can be analyzed as shown in the last layer of Figure 4. An example query using the SPARQL query language is

⁶ <http://ontorule-project.eu/resources/assembler/process-ontology-and-facts.owl>

LISTING 1: Excerpt of the RDF Serialization of the Use Case Example - namespaces were neglected

```

<rdf:Description rdf:about= "NS#Forward all forms to IT department for scanning" >
  <rdf:type rdf:resource= "NS#Task" />
  <follows rdf:resource= "NS#Perform final check of information" />
  <hasExecutionTime rdf:datatype="http://www.w3.org/2001/XMLSchema#string" >
    | 00:000:00:03:00
  </hasExecutionTime>
</rdf:Description>
<rdf:Description rdf:about= "NS#Perform final check of information" >
  <rdf:type rdf:resource= "NS#Task" />
  <follows rdf:resource= "NS#Forward remaining forms to inspection department" />
  <hasExecutionTime rdf:datatype="http://www.w3.org/2001/XMLSchema#string" >
    | 00:000:00:02:00
  </hasExecutionTime>
</rdf:Description>

```

shown in Listing 2. This query selects all predecessor activities of the *Perform final check of information* activity.

LISTING 2: Example of SPARQL Query on an RDF Representation of a Model from the Use Case

```

SELECT ?predecessor WHERE {
  ?predecessor ns:follows ns:Perform final check of information
}

```

Result: ns:Forward all forms to IT department for scanning

6 Discussion

With the technical realization of the approach and its application to a use case we can conclude that the presented approach is useful for semantically enriching existing visual models ex-post. Thereby, neither the models nor the used ontologies have to be modified with the creation of annotations. Thus, the annotations are not limited by the type of model or ontology. However, in the described implementation, the ontologies have to be imported as visual models in order to use our annotation approach. The described weaving approach requires however that the modeling tool used for it supports model references. Hence, tools which do not support model references have to be adapted to realize the loosely coupled semantic annotation approach. A performance analysis is part of our further research.

In summary we can derive a number of benefits as well as also some drawbacks of the approach in its current version. These are listed in Table 2.

Benefits	Drawbacks
⊕ Customization of RDF generation with visual annotations	⊖ Visual annotations may become complex to handle
⊕ Independent of the used modeling language for conceptual models	⊖ Direct annotation references to ontology concepts not implemented yet
⊕ Types of annotations are extendable	⊖ Semantics of annotation types needs to be provided separately via rules
⊕ Annotations are re-usable	⊖ Ontology schema concepts are required
⊕ OWL import/export options exist	⊖ RDF-serialization for non-OWL ontologies not implemented yet

Tab. 2: Benefits and drawbacks of the introduced approach

Usability test and economical analysis are two aspects which are out of the scope of this paper but which have to be done before implementing the approach in industry modeling tools. The linking mechanism - from the visual annotation to model elements - is probably the most challenging feature. This is because the linking mechanism has to be generic so that model elements created with different modeling languages and tools can be referenced. Further, we see the support of different ontologies - as described in Table 2 - as an important feature to make the approach feasible.

The introduced approach enables institutions to semantically enrich their existing models created with different modeling tools. Hence, they save costs for remodelling and standardizing the existing models. However, the introduced approach requires human-created annotations so that institutions face a trade-off between costs for remodelling and costs for creating semantic annotations.

7 Conclusion and Further Research

In this paper we introduced a model weaving approach for transforming conceptual models to RDF. For this purpose we introduced a visual modeling language for creating model annotations. The annotations are neither limited to a specific kind of conceptual model nor to a specific kind of ontology. The technical feasibility of the approach has been shown by implementing it on the SeMFIS platform and applying it to a use case.

In our future research we want to develop further types of annotations and introduce ontology references which point directly to ontology schema constructs, e.g. as contained in an ontology repository. In this way the transformation of ontologies to visual models could be omitted. Another aspect that will be investigated will be the usability of the approach. For this purpose especially the procedures of annotating existing models will have to be tested with users to judge whether the used modeling language is adequate in a practical setting. Economical as well as usability analysis are part of our further research.

References

- [APW08] Awad, Ahmed; Polyvyanyy, Artem; Weske, Mathias: Semantic querying of business process models. In: Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE. IEEE, pp. 85–94, 2008.
- [BB12] Belghiat, Aissam; Bourahla, Mustapha: Transformation of UML models towards OWL ontologies. In: Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on. IEEE, pp. 840–846, 2012.
- [BHB09] Bizer, Christian; Heath, Tom; Berners-Lee, Tim: Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
- [BK15] Buchmann, Robert Andrei; Karagiannis, Dimitris: Pattern-based Transformation of Diagrammatic Conceptual Models for Semantic Enrichment in the Web of Data. In: 19th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2015, Singapore, 7-9 September 2015. pp. 150–159, 2015.
- [BK16] Buchmann, Robert A.; Karagiannis, Dimitris: Enriching Linked Data with Semantics from Domain-Specific Diagrammatic Models. *Business & Information Systems Engineering*, 58(5):341–353, 2016.
- [Cr01] Cranefield, Stephen: Networked Knowledge Representation and Exchange using UML and RDF. *J. Digit. Inf.*, 1(8), 2001.
- [DFBV06] Del Fabro, M. Didonet; Bézivin, Jean; Valduriez, Patrick: Weaving Models with the Eclipse AMW plugin. In: Eclipse Modeling Symposium, Eclipse Summit Europe. volume 2006, 2006.
- [DFJ05] Del Fabro, Marcos Didonet; Jouault, Frédéric: Model transformation and weaving in the AMMA platform. *Proceedings of GTTSE*, 2006, 2005.
- [EKO07] Ehrig, Marc; Koschmider, Agnes; Oberweis, Andreas: Measuring similarity between semantic business process models. In: Proceedings of the fourth Asia-Pacific conference on Conceptual modelling-Volume 67. Australian Computer Society, Inc., pp. 71–80, 2007.
- [Fi11] Fill, Hans-Georg: On the Conceptualization of a Modeling Language for Semantic Model Annotations. In: Advanced Information Systems Engineering Workshops - CAiSE 2011 International Workshops, London, UK, June 20-24, 2011. *Proceedings*. pp. 134–148, 2011.
- [Fi12] Fill, Hans-Georg: An Approach for Analyzing the Effects of Risks on Business Processes using Semantic Annotations. In: 20th European Conference on Information Systems, ECIS 2012, Barcelona, Spain, June 10-13, 2012. p. 111, 2012.
- [Fi17] Fill, Hans-Georg: SeMFIS: A flexible engineering platform for semantic annotations of conceptual models. *Semantic Web*, 8(5):747–763, 2017.
- [FRK12] Fill, Hans-Georg; Redmond, Timothy; Karagiannis, Dimitris: FDMM: A Formalism for Describing ADOxx Meta Models and Models. In: ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, Volume 3, Wroclaw, Poland, 28 June - 1 July, 2012. pp. 133–144, 2012.

- [Ga04] Gasevic, Dragan; Djuric, Dragan; Devedzic, Vladan; Damjanovic, Violeta: Converting UML to OWL ontologies. In: Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004. pp. 488–489, 2004.
- [Hi15] Hinkelmann, Knut: Modeling Framework for BPaaS. CloudSocket, December 2015. <https://www.cloudsocket.eu/documents/10182/20690/CloudSocket-D3.1-BPaaS+Design+Environment+Research/91a3c2ae-6394-482a-940e-d0186e82f7f6>, Accessed on 13-04-2017.
- [Jo06] Jouault, Frédéric; Allilaire, Freddy; Bézivin, Jean; Kurtev, Ivan; Valduriez, Patrick: ATL: a QVT-like transformation language. In: ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. ACM, pp. 719–720, 2006.
- [Ka06] Kappel, Gerti; Kapsammer, Elisabeth; Kargl, Horst; Kramler, Gerhard; Reiter, Thomas; Retschitzegger, Werner; Schwinger, Wieland; Wimmer, Manuel: Lifting metamodels to ontologies: A step to the semantic integration of modeling languages. In: International Conference on Model Driven Engineering Languages and Systems. Springer, pp. 528–542, 2006.
- [KB16] Karagiannis, Dimitris; Buchmann, Robert Andrei: Linked Open Models: Extending Linked Open Data with conceptual model information. *Inf. Syst.*, 56:174–197, 2016.
- [KJS96] Karagiannis, Dimitris; Junginger, Stefan; Strobl, Robert: Introduction to Business Process Management Systems Concepts. In: Business process modelling, pp. 81–106. Springer, 1996.
- [KMM16] Karagiannis, Dimitris; Mayr, Heinrich C.; Mylopoulos, John, eds. Domain-Specific Conceptual Modeling, Concepts, Methods and Tools. Springer, 2016.
- [My92] Mylopoulos, John: Conceptual modelling and Telos. *Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development*, New York: John Wiley & Sons, pp. 49–68, 1992.
- [Ro06] Rosemann, Michael: Potential pitfalls of process modeling: part A. *Business Process Management Journal*, 12(2):249–254, 2006.
- [Sc08] Schätz, Bernhard: Formalization and rule-based transformation of EMF Ecore-based models. In: International Conference on Software Language Engineering. Springer, pp. 227–244, 2008.
- [TF07] Thomas, Oliver; Fellmann, Michael: Semantic EPC: Enhancing Process Modeling Using Ontology Languages. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007, held in conjunction with the 3rd European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, June 7, 2007. 2007.
- [vDDM13] van Dongen, Boudewijn F.; Dijkman, Remco M.; Mendling, Jan: Measuring Similarity between Business Process Models. In: *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, pp. 405–419. 2013.
- [WH01] Whitman, Larry; Huff, Brian: On the Use of Enterprise Models. *International Journal of Flexible Manufacturing Systems*, 13(2), 2001.