Supporting Process Model Development with Enterprise-Specific Ontologies

Nadejda Alkhaldi[1], Sven Casteleyn[1,2] and Frederik Gailly[3]
[1]*Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium*
[2]*Universitat Jaume I, Av. De Vicent Sos Baynat s/n, 12071 Castellon, Spain*
[3]*Ghent University, Tweekerkenstaart 2, 9000 Gent, Belgium*

Keywords: Enterprise Modelling, Enterprise Ontology, Ontology-driven Modelling, BPMN.

Abstract: Within an enterprise, different models – even of the same type - are typically created by different modellers. Those models use different terminology, are based on different semantics and are thus hard to integrate. A possible solution is using an enterprise-specific ontology as a reference during model creation. This allows basing all the models created within one enterprise upon a shared semantic repository, mitigating the need for model integration and promoting interoperability. The challenge here is that the enterprise-specific ontology can be very extensive, making it hard for the modeller to select the appropriate ontology concepts to associate with model elements. In this paper we focus on process modelling, and develop a method that uses four different matching mechanisms to suggest the most relevant enterprise-specific ontology concepts to the modeller while he is creating the model. The first two utilize string and semantic matching techniques (i.e., synonyms) to compare the BPMN construct's label with enterprise-specific ontology concepts. The other two exploit the formally defined grounding of the enterprise ontology in a core ontology to make suggestions, based on the BPMN construct type and the relative position (in the model). We show how our method leads to semantically annotated process models, and demonstrate it using an ontology in the financial domain.

1 INTRODUCTION

When different models within an enterprise are created by different modellers, integrating those models is hard. This is known as the correspondence problem (Fox and Grüninger, 1997). A possible solution for this integration problem is providing modellers with a shared vocabulary formalized in an ontology (Bera et al., 2009). Over the last 30 years, different domain ontologies have been developed which describe the concepts, relations between concepts and axioms of a specific domain. In a business context, a particular type of domain ontologies is so-called enterprise ontologies. They describe the enterprise domain and consequently provide enterprise domain concepts that can be reused by different enterprises. Example of enterprise ontologies include the Enterprise Ontology (Uschold et al., 1998), TOVE (Fox, 1992) and the Resource Event Agent enterprise ontology (Gailly et al., 2008). Two different approaches have been proposed to incorporate enterprise ontologies into the modelling process. Some authors consider enterprise ontologies to be reference models that support the creation of different kind of models. For instance, (Fox and Grüninger, 1997) suggests developing the Generic Enterprise Model as an ontology that is later used as a reference for creating both data and process models. Other authors developed an enterprise-specific modelling language which is based on the concepts, relations and axioms described in the enterprise ontology (Sonnenberg et al., 2011). There are also works proposing a process ontology to be used specifically for process models, e.g., (Klein and Bernstein, 2001), (Schlenoff et al., 1998) and (Haller et al., 2008). These works are mainly concentrating on constructing those ontologies, rather than supporting the modeller that needs to use them.

In this paper we focus on using enterprise-specific ontologies (ESO) during the development of business process models. Enterprise-specific ontologies are domain ontologies that differ from enterprise ontologies in the fact that their Universe of Discourse is a specific enterprise, rather than the enterprise domain. They may have their origin in an established domain ontology or in an enterprise ontology, but their main goal is describing the concepts, relations and axioms that are shared within a particular enterprise. Enterprise-specific ontologies are getting increasingly important in the context of Data Governance and knowledge representation (Bera et al., 2011). Supporting tools, such as IBM InfoSphere[1] or Collibra Enterprise Glossary[2] allow enterprises to specify their own enterprise glossary/ontology. Such an enterprise-specific ontology, once available, can subsequently be deployed to help enterprise modellers in creating compatible, enterprise-specific models, such as requirements, data or process models. This paper focuses on business process models, and presents mechanisms for assisting the business process modeller by suggesting relevant terms while he is constructing models, based on the enterprise-specific ontology. Indeed, enterprise workers sometimes do not know which knowledge they need to perform their work (Bera et al., 2011), do not know all the concepts they need to make their models and/or do not know the agreed upon terminology. By providing a limited set of relevant concepts to the modeller, originating from the enterprise-specific ontology, we effectively help him in his modelling task, while promoting cross-model reuse of existing concepts. The use of the enterprise-specific ontology furthermore allows to semantically annotate modelling elements in the resulting models, thereby ensuring compatibility and integrateability of different models.

The work described in this paper is part of a larger framework for ontology-driven enterprise modelling aimed to facilitate model construction based on an enterprise-specific ontology on one hand, and support enterprise-specific ontology creation and evolution based on feedback from the modelling process on the other hand. The framework thus explores the symbiotic relationship between the models and the ontology. This paper elaborates two phases of the framework, namely the enterprise-ontology-based model suggestion mechanisms, and the model creation process. Other phases are not elaborated here. To illustrate our work, we use the Unified Foundational Ontology as core ontology, OWL as ontology representation language and BPMN as business process modelling language.

[1] http://www-01.ibm.com/software/data/infosphere/

[2] http://www.collibra.com/

2 ENTERPRISE-SPECIFIC ONTOLOGY AND MODELLING META-METHOD

As displayed in Figure 1, the previously outlined framework (i.e., a meta-model) consists of two parallel cycles: an ontology engineering cycle and enterprise modelling cycle. Before a model can be created, the enterprise-specific ontology needs to be set up (Ontology Setup Phase in Figure 1). More specifically, if not yet previously done, the concepts and relationships contained in the enterprise-specific ontology[3] (ESO) are analysed using a core ontology. A core ontology is an ontology that describes universally agreed upon, high level concepts and relations, such as objects, events, agents, etc. Grounding the enterprise-specific ontology in a core ontology provides well-founded semantics, and facilitates the modelling suggestion mechanisms (see further). Once the enterprise-specific ontology is set up, the enterprise modelling cycle starts with the modeller selecting a model type (Enterprise Model Selection phase), such as i*, ER, BPMN, Petri net, etc. In our method, modelling languages that were previously analysed using the same core ontology as the ESO are particularly useful, as these facilitate particular modelling suggestions based on the enterprise-specific ontology, and using the core ontology as an intermediary (see section 3). While creating constructs in the model (Enterprise Model Creation phase), aided by suggestions generated based upon the ESO (Ontology Storage and Suggestion Generation phase), the model is automatically annotated with ESO concepts/relations, and the modeller reports feedback on ontology content: missing concepts or relations, or inaccurate ones. When the model is finished, model quality is evaluated (Enterprise Model Evaluation phase) and once it satisfies certain quality standards, the reported feedback is presented to the enterprise stakeholders (i.e., analysts, modellers, developers, managers, etc.) for discussion (Community-based Ontology Feedback Evaluation phase). If this community decides the feedback is valuable within the context of the enterprise, it is added to the enterprise-specific ontology (Ontology Evolution phase), thus representing the evolution of the ontology in a new version that corresponds better to the business reality of the enterprise.

The proposed meta-method is unique because we

[3] If no ESO is available, it can either be constructed from scratch, or an existing domain ontology covering the business domain of the enterprise may serve as a starting point.

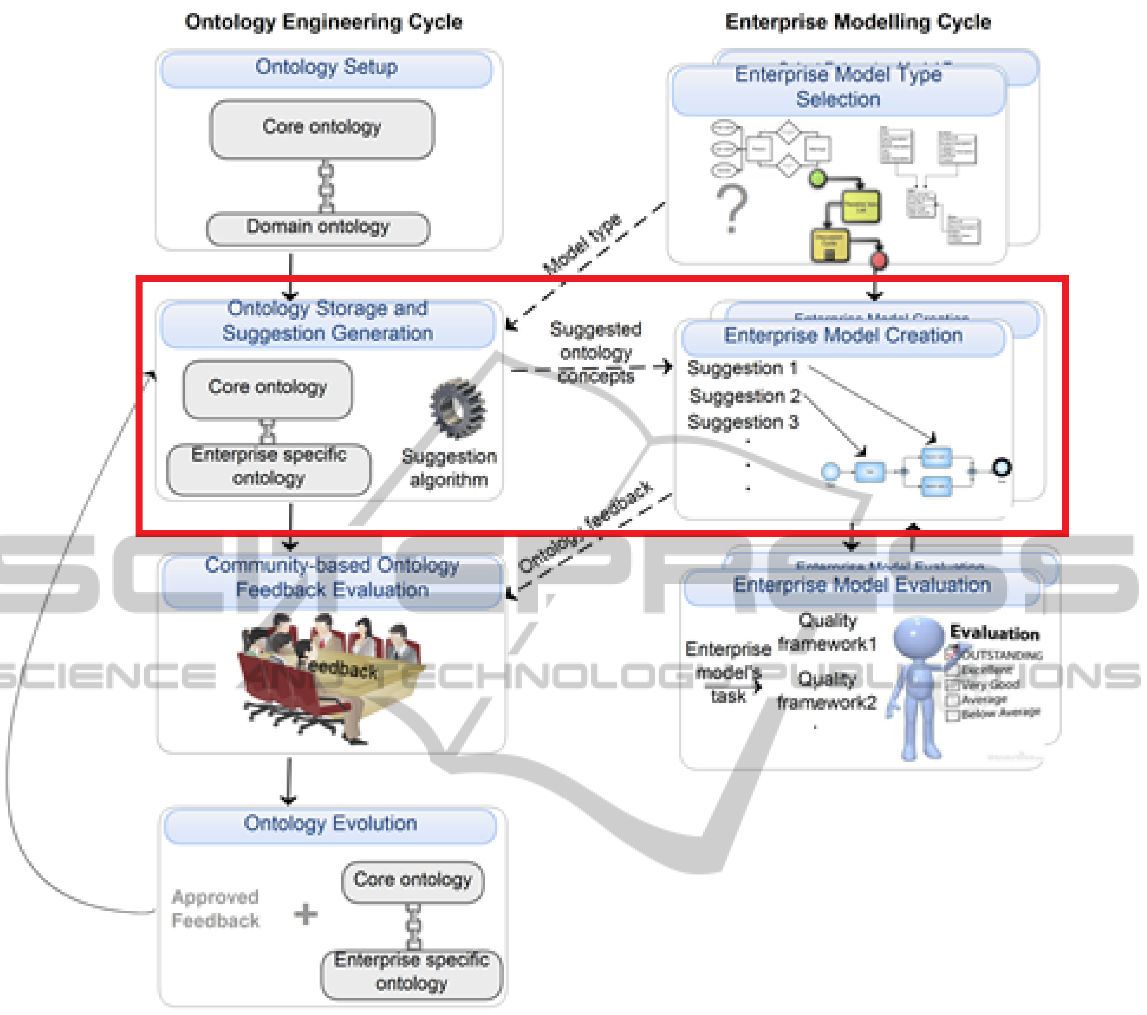

Figure 1: Enterprise-specific ontology and modelling meta-method.

do not focus on a specific modelling language, and we furthermore aim to exploit the symbiotic relationship between the enterprise modelling cycle and the ontology engineering cycle. Nevertheless, several researchers have already performed research isolated in a single phase of the meta-method. Relevant for this article, several researchers have investigated how business process models can be semantically annotated using ontologies. (Liao et al., 2013) describe how processes and subprocesses can be annotaed by sets of ontology concepts. In (Lin and Krogstie, 2010) the authors perform different kinds of annotation on process models, including profile, meta-model, model and goal annotation. Of these annotations, model annotations are immediately relevant for our work, which are stored in a Process Semantic Annotation Model, along with the elements of that model. In both these works, the proposed method seems to be manual, and does not focus on providing suggestions while the model is under construction. Next to business process model annotation, different researchers have also investigated how natural Language Processing techniques can be used while creating business process models. For an overview we refer to (Leopold, 2013). The focus here mainly lies with extracting information and adding semantic annotations to exiting models, rather than assisting the modeller during modelling as in our work.

This paper focuses on the second phase of both the ontology engineering and enterprise modelling cycles. More specifically, it explains the mechanisms that provide suggestion for the labels of modelling construct suggestions to the modeller, based on the enterprise-specific ontology, and details how models are created and annotated with enterprise-specific concepts.

3 SUGGESTION GENERATION MECHANISMS

Our suggestion generation algorithm returns a customized suggestion list of ESO concepts for (the label of) every BPMN construct selected by the modeller and placed on the canvas. Given the potentially extensive amount of ESO concepts, relevance ranking of suggestions is a critical feature. Depending on the type of modelling construct that is added, the position of the construct relevant to other elements (i.e., its neighbourhood) in the model, and the label entered by the modeller, the order of the suggestion list is prioritized so that ontology concepts with a higher likelihood to be relevant in the current context appear first. To achieve this, four different suggestion generation mechanisms are used (Figure 2). These mechanisms are partly inspired by ontology matching techniques, (Euzenat and Shvaiko, 2013) but are specifically focused to fit within our framework, where the semantics of the modelling language (i.e., BPMN) meta-model can be exploited. The first mechanism is the use of string matching based on the BPMN element's label (partially or in full entered by the modeller). The second is synonym matching, which retrieves synonyms of the BPMN element's label and verifies whether there are syntactically equivalent concepts in the ESO. The third mechanism derives suggestions based on the relation between the constructs of the modelling language (i.e. BPMN) and the ESO concepts, using the core ontology (i.e., UFO) as intermediary. We call this "construct matching". The last suggestion generation mechanism derives suggestions from the position of a given BPMN construct relative to other elements in the BPMN model. We call this "neighbourhood-based matching".

Every matching technique calculates a relevance score (between 0 and 1) for each ESO concept, which is stored. Subsequently, the overall relevance score is calculated using a weighted average of all individual scores. This corresponds to the formula below:

$$ConceptRelevanceScore = \frac{S_s W_s + S_{syn} W_{syn} + S_c W_c + S_l W_l}{W_s + W_{syn} + W_c + W_l}$$

Where:

$S_s W_s$: the score and weight of string match
$S_{syn} W_{syn}$: the score and weight of synonym match
$S_c W_c$: the score and weight of construct match
$S_l W_l$: the score and weight of neighbourhood based match

The weights for each matching mechanism are thus configurable. In our demonstration (see section 4), we assigned a higher weight to string matching as we expect that, within a particular enterprise context, a (quasi) exact string match has a high possibility of representing the intended (semantic) concept. The lowest weight is assigned to construct matching, because it typically matches very broadly, and thus delivers a large amount of suggestions. Further experimentation with the distribution of weights over the individual relevance scores should be performed to determine an optimal overall score calculation. This is considered future work. As a final result, the suggestion list, a descending ordered list of ESO concepts according to relevance, is generated and presented to the modeller. In the next four subsections the different mechanisms are described in more detail. The implementation of the mechanisms can be consulted via our Github repository: https://github.ugent.be/MIS .

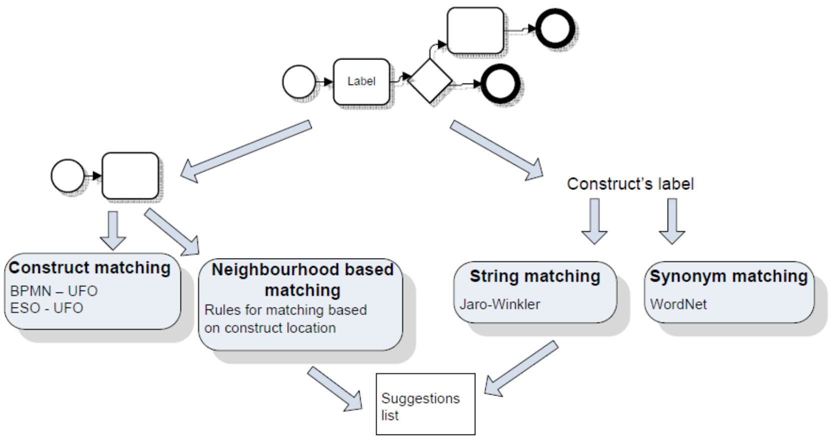

Figure 2: Suggestion generation algorithms.

final result, the suggestion list, a descending ordered list of ESO concepts according to relevance, is generated and presented to the modeller. In the next four subsections the different mechanisms are described in more detail. The implementation of the mechanisms can be consulted via our Github repository: [URL suppressed for blind review].

3.1 String Matching Mechanism

The goal of the string matching algorithm is to find ESO concepts whose label is syntactically similar to the label of BPMN elements entered by the modeller. If these two strings are syntactically the same, there is a high possibility that they have the same semantics, especially as both reside within the same enterprise and business context.

The string matching mechanism thus finds ESO matches based on the text entered by the modeller as label for a modelling (i.e., BPMN) element. There are many ways to compare strings depending on whether the string is seen as an exact sequence of letters, an erroneous sequence of letters or a substring of a set of words. Currently we use the Jaro-Winkler distance (Winkler, 1990) to calculate the edit distance between the given BPMN element label, and the label of each concept in the enterprise-specific ontology. The Jaro-Winkler distance was chosen because this hybrid technique takes into account that the text entered by the modeller can contain spelling errors, and additionally favours matches between strings with longer common prefixes (i.e., a substring test, which is very useful in our context because matching is executed each time a character is added to the label).

Jaro-Winkler distances are between 0 (no similarity) and 1 (exact match), and are thus immediately useable as a relevance score. For some modelling constructs, only a part of the label is matched. For example, tasks' names in process models are often specified as a <verb> <noun> combination (Dumas, La Rosa, Mendling and Reijers, 2013), and our method only seeks to find matching for the <noun> part of the label, as only this part can correspond to a concept in the ontology. In general, for the following BPMN constructs the full label is being matched: pool, lane, message flow and data object. Conversely, for the following BPMN constructs, only partial matching is performed: task, sub-process, event and conditional gateway. The labels of these constructs are analysed using standard natural language parsing techniques, which allow identifying the nouns within the labels.

3.2 Synonym Matching Mechanism

The synonym matching mechanism aims to detect synonyms of the given BPMN element label (or part of it) in the ESO. To realize this, we use WordNet (Miller, 1995), an online lexical database that organises English nouns, verbs, adjectives and adverbs into sets of synonyms (so-called synsets). It is thus ideal to find synonyms. For each synonym of the BPMN element label, the previously described string matching algorithm is performed on all ESO concepts, thereby generating a relevance score between 0 (no match) and 1 (exact synonym match). Synonym matching allows the modeller to find concepts that are semantically similar, but for which he used a different label. For example, while the modeller enters the label "customer", the concept "client" might be available in the enterprise-specific ontology, and should be given as a suggestion. Synonyms are semantically equivalent words, but their usage depends on a certain domain. Therefore, this mechanism may generate false positives, in case synonyms in a different context/domain are matched. As we are operating within one particular domain, namely that of the enterprise, the chance of false positives is low.

3.3 Construct Matching Mechanism

The construct matching algorithm finds ESO suggestions based on the type of modelling constructs the modeller choses. To do so, the algorithm exploits the grounding of (i.e., mappings between) ESO concepts and a core ontology on one hand, and mappings between the constructs of the modelling language (i.e., BPMN) meta-model and the (same) core ontology on the other hand. The core ontology thus acts as an intermediary between the ESO and (the meta-model of) the chosen modelling language. The core ontology that we used in this paper is the Unified Foundational Ontology (UFO), for three reasons: 1/ the benefits of grounding domain ontologies in UFO are well motivated (Guizzardi, Falbo and Guizzardi, 2008), and several such UFO-grounded domain ontologies are available, e.g., (Barcellos, Falbo and Moro, 2010) 2/ UFO is specifically developed for the ontological analysis of modelling languages, and 3/ an analysis of BPMN using UFO is available in literature (Guizzardi and Wagner, 2011), and can thus be re-used. No claim is made here that UFO alone is sufficient to find the match. However, using UFO's predefined categories, according to which ESO concepts and BPMN constructs are classified, allows to guide construct-based matching and narrow down the relevant suggestions. UFO has different layers of which we here only use UFO-U (represented in Figure 2), as this is sufficient for finding construct-based matches between BPMN and ESO. Our demonstration supports this claim; however, it can be further investigated if using (full) UFO is beneficial to refine our algorithm. Additionally, other modelling languages might require more specialized UFO concepts, in which case (full) UFO will be needed. The top level element in UFO-U is a Universal. It represents a classifier that classifies at any moment of time a set of real world individuals and can be of four kinds: Event type, Quality universal, Relator universal and Object type.

Grounding the enterprise ontology in the UFO ontology is either done manually, as shown in our demonstration in section 5, or given when an existing domain ontology that was previously grounded, is re-used as initial enterprise ontology. This effort is thus domain- or even enterprise-specific. On the other hand, the mapping between the modelling language's meta-model and the UFO ontology is generally applicable, and several mappings are available in literature, e.g., UML (Guizzardi and Wagner, 2008), BPMN (Guizzardi and Wagner, 2011).

We can thus re-use the mapping provided by (Guizzardi and Wagner, 2011) for UFO - BPMN. However, in (Guizzardi and Wagner, 2011), the aim was to perform an in depth ontological analysis of BPMN, thus providing the mapping between UFO and BPMN. As we limited the used core ontology to UFO-U, which is sufficient to find matches between BPMN and the ESO, we need to adjust the mappings provided by (Guizzardi and Wagner, 2011). Once

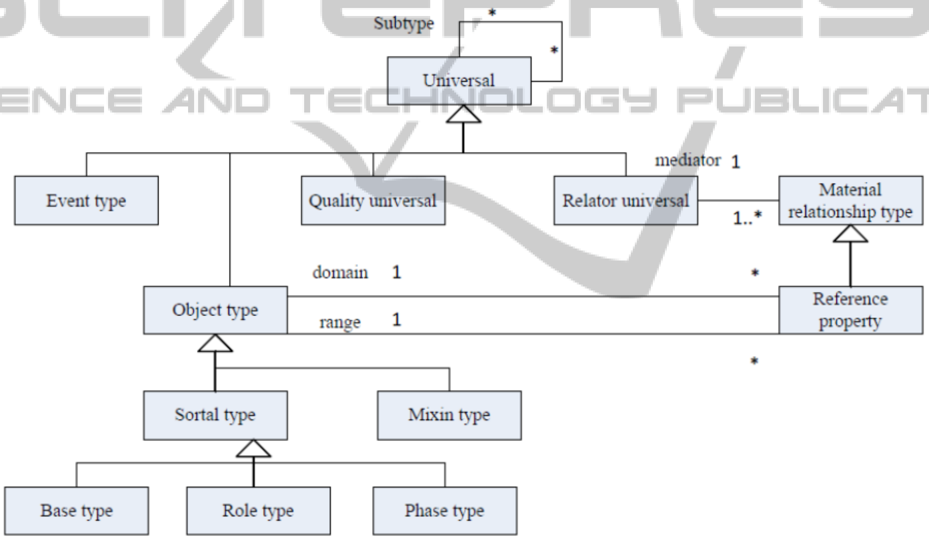

Figure 3: UFO-U.

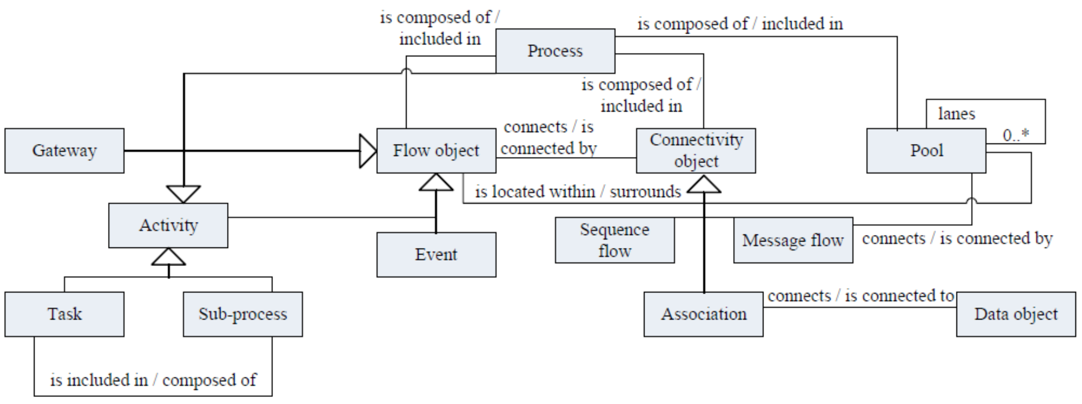
is composed of / included in
Process
is composed of / included in
is composed of / included in
lanes
0..*
Gateway
Flow object
connects / is connected by
Connectivity object
Pool
is located within / surrounds
Activity
Event
Sequence flow
Message flow
connects / is connected by
Task
Sub-process
Association
connects / is connected to
Data object
is included in / composed of

Figure 4: BPMN meta-model.

Table 1: Mappings between BPMN constructs and UFO-U.

BPMN construct	**UFO-U**	**BPMN construct**	**UFO-U**
Pool	Object type	End event	Event Type
Lane	Object type	Event noun	Base type, Mixin type, Relator universal
Task Noun	Relator universals or quality universal	Condition Exclusive Gateway	Quality universal
Noun Sub Process	Relator universals or quality universal	Data object	Relator universal,, Base type
Start event	Event Type	Message flow label	Relator universal
Intermediate event	Event Type		

given, this adjusted mapping is re-usable by any subsequent BPMN modelling efforts. The BPMN meta-model that is used in this paper is shown in Figure 4 and is based on the original OMG BPMN standard.

Table 1 represents the mappings between the constructs of the BPMN meta-model and UFO-U. Limiting the match to UFO-U concepts means that for instance both the Pool and Lane constructs are mapped to UFO Object types whereas in (Guizzardi and Wagner, 2011) they are mapped to Agents, which is a special type of UFO-U Object type. In (Guizzardi and Wagner, 2011) a task is mapped to an Action Event Types, which is a special kind of Event. In this paper we extend this mapping based on the observation that a lot of well-known BPMN textbooks such as (Dumas, La Rosa, Mendling and Reijers, 2013) suggests BPMN modellers to follow the pattern "verb noun" when specifying the name of task. The noun that is used corresponds to a UFO-U relator or quality universal. The mappings of a subprocess are identical to those of the Task construct.

The different types of BPMN Events are all mapped to UFO-U Event Type. It is important to notice that in most cases the ESO will not contain UFO-U events, as ESOs most often focus on the static part of the vocabulary of the enterprise, i.e., similar to the view on domain specific ontologies in (Lin et al., 2006). However, for every BPMN model, we can search for UFO-U objects that participate in these events. Gateways are not mapped directly to UFO-U concepts but for an exclusive data-based gateway we can expect that the condition that is evaluated will use properties of UFO Object Types or UFO Relator Types which in UFO-U correspond to qualities. Data objects and message flow objects are most likely to be relators, such as contract, or base types such as technical documentation of some software. The association and sequence flow connectivity objects of the BPMN meta-model are not mapped to UFO-U, but they will be used extensively in the construct neighbourhood-based mechanism (see section 3.4).

To formalize these mappings, we first transformed both UFO-U and the BPMN meta-model into OWL ontologies [4], using the transformations provided by OMG's Ontology Definition Meta-model (OMG, 2006), and subsequently formalized the mappings between the constructs of these ontologies using a third OWL ontology[5]. Using OWL for both representing the ontologies and the mappings has some advantages: 1) OWL supports different approaches for specifying the mapping (OWL equivalent classes, SWRL rules), 2) OWL supports reasoning which will be used to generate the list of suggestions based on the provided mappings and 3) OWL uses URIs to identify the concepts which will be used during the annotations of the models.

The actual mappings in the OWL mapping ontology are specified using the OWL equivalent class construct. The mapping ontology consists of the UFO-U to BPMN mappings and the UFO-U to ESO mappings, both of which are specified separately and subsequently imported. Next, a reasoner is used to create an inferred class hierarchy of ESO concepts that are matched to a given BPMN construct. For instance, when a BPMN pool is created the inferred class hierarchy is used to determine all subclasses of BPMN pool (i.e., UFO-U concepts). Subsequently, all ESO concepts related to these subclasses (using the UFO-U to ESO mapping) will receive the relevance score of 1.0, while all the other ESO concepts will be considered irrelevant and will receive a relevance score of 0. This mechanism has been implemented using the OWL Java API and is available from the GitHub repository.

[4] Download from: https://github.ugent.be/MIS/Ontology BasedSuggestionAlgorithms
[5] Download from: https://github.ugent.be/MIS/ Ontology BasedSuggestionAlgorithms

3.4 Neighbourhood-based Mechanism

Before the element neighbourhood-based mechanism is described, we first explain how ontology annotations are stored, as existing annotations are used by this mechanism. In order not to jeopardize the re-usability of our suggestion generation mechanism among different tools, we choose not to extend the BPMN meta-model with annotation elements. Instead, we opted to create, for every BPMN model, an OWL ontology that contains instantiations of the BPMN OWL classes. When a modeller decides to annotate a newly created element with an ESO concept, a URI of the corresponding OWL ESO concept is added to the OWL BPMN individual using the OWL annotation mechanism. In this way, it is always possible to refer from the BPMN model elements to the reference ESO concepts.

With the annotation mechanism explained, we now detail the element neighbourhood-based mechanism, which calculates relevance scores for ESO concepts based on the location of the BPMN element that is added to the model, the type of construct that is added, and the relationships between the ESO concepts. The neighbourhood of a BPMN element is determined by the connectivity objects (i.e. sequence flow, message flow, association), and the lanes recursive relationship of the BPMN meta-model. In other words, for every element we can determine which pool or lane it is a part of, and which other element(s) is/are connected to this element using either a sequence, message flow or association. Next, the relationships (which are specified in terms of the UFO-U relationships through the ESO-UFO-U mappings) between the ESO concepts are exploited. According to (Guizzardi and Wagner, 2008) there are two types of relations: formal and material. *Formal relation* holds between entities directly without any further intervening individuals. *Material relation* has material structure on its own. Entities related by this type of relation are mediated by individuals called relators. In section 4 we will demonstrate how the UFO-U relators can be used to specify the material relations between the ESO concepts.

Finally, using both the relative position of the new element and the material relations between the ESO concepts, the element neighbourhood-based mechanism can now derive relevance scores for ESO concepts in relation to some BPMN model construct (for examples, see section 4):

1. To create a pool construct when another pool already exists, the suggestions (relevance score 1) are UFO-U object types that are related by a material relationship with the ESO concept with which the existing pool(s) was/were annotated (i.e., the ontology annotations of the pool(s)).
2. To create a lane construct within a pool, the suggestions (relevance score 1) are UFO-U object types that are related by a material relationship with the ontology annotation of the pool(s)
3. To create a message construct that results in transmitting a message between a task or event of a pool and another pool, the suggestions (relevance score 1) are UFO-U relators mediating material relations connecting objects that annotate respectively the noun of the task and the ontology annotation of the pool.
4. To create a conditional gateway, there are two ways to derive suggestions (both receive relevance score 1):
   - ESO concepts annotated by the task label preceding the gateway. This can work very well for tasks that are performing evaluation or calculation, after which the gateway is used to make a decision based on the results. In this case, the condition on the gateway will use the same concept as used in the task. This concept is most likely to be a quality, especially if the task at hand is performing calculations. Nevertheless, it can also be a relator, such as for example verifying if the contract is ok or not.
   - Qualities associated with the UFO-U object type annotation of the pool where the gateway is located. Or UFO-U qualities associated with UFO-U object types participating in material relations with Object type annotation of the pool where the gateway is located.
5. For creation of a task construct, the suggested concepts are most likely to be related through material relations to the pool where the task is located. The suggestions can be either object types or relators mediating those material relations.

The implementation of this mechanism is more complex because the algorithm needs to not only know the location (relative to other elements) of the new element but also the annotations of the modelling elements to which the modeller plans to connect the new element. As mentioned in the beginning of this section, an OWL ontology file is created for every model; it contains individuals that instantiate the BPMN OWL classes. This file, in particular the instantiations of connectivity concepts, is used to determine the position of the newly created element. Next for selected connected element (i.e., those satisfying one of the previous 5 rules) the algorithms checks whether it contains an ESO ontological annotation. In case the surrounding element is annotated, the algorithm uses a SPARQL query to identify to which ESO concepts the ontological annotation is related. The ESO concepts that are returned by this query consequently receive a weight of 1.

4 DEMONSTRATION

This section illustrates the method explained in the previous sections by means of a lab demonstration in which the modeller constructs a process model in the

Table 2: Mappings between ESO concepts and UFO.

ESO concept	UFO-U	ESO concept	UFO-U
AddedValue	Quality_Universal	Liability	Relator Universal Mediates Customer and mediates Branch
Adminstrative	Role_Type	Loan	Relator Universal Mediates some X and mediates some X
Asset	Mixin Type	MortgageLoan	Relator Universal Mediates some X and mediates some X
Branch	Base_Type	Payment	Relator Universal Mediates some X and mediates some X
Channel	Relator Universal Mediates some Customer and mediates some Bank	Person	Base_Type
Collection	Quality_Universal	Product	Mixin_Type
Commercial	Role_Type	ProductRateApplication	Quality_Universal
Company	Base_Type	ProductRateApplication Fixed	Quality_Universal
Corporative	Base_Type	ProductRateApplication Fixed	Quality_Universal
Currency	Base_Type	ProductRateApplication Variable	Quality_Universal
CurrentMortgageLoan	Relator Universal Mediates some X and mediates some X	Quota	Quality_Universal
Customer	Mixin_Type	SME	Base_Type
Department	Base_Type	SOHO	Base_Type
Employee	Role_Type	SavingsAccount	Base_Type
FutureMortgageLoan	Relator Universal Mediates some X and mediates some X	Service	Mixin_Type
Individuals	Base_Type	ServiceContractBy Customer in Chanel	Mixin_Type
InvestmentAccount	Base_Type	Staff	Role_Type
InvestmentFund	Base_Type	User	Mixin_Type
Invoice	Relator Universal Mediates some X and mediates some X	vBanking	Base_Type

financial domain using an existing financial domain ontology [6] as enterprise-specific ontology. This ontology contains static concepts related to finance, such as Branch, Customer, Loan, Insurance, etc., which can be used as a reference for models constructed in the financial domain. As a first step, the ESO needs to be mapped to the UFO-U core ontology. This classification is part of the ontology engineering cycle of our method and is performed during the ontology setup phase. Table 2 represents the mapping, which is formalized in OWL and can be downloaded from our repository. Important to notice is that in our mapping, when an ESO concept is classified as a Relator we also link this Relator to the Object Types it connects. For example, a relator Loan is relating Branch and Customer entities. The ESO also contains OWL data properties, which are all considered to be quality universals because they depend on the universals they are related to. For example, Expiration Date of a Product is a quality universal because it depends on the universal Product.

Once the ESO is grounded in the core ontology (as mentioned, this only needs to be done once, and can subsequently be re-used for any model created within the enterprise), the modeller can start creating the process model. He selects a construct to be added, places it on the canvas and starts typing the construct's desired name. As he selects the construct, and as he is typing, the mechanisms described in the previous section derive suggestions from the ESO and present them to the modeller. If an ESO concept in the suggestion list appropriately corresponds to the intention of the modeller for this particular BPMN construct, he selects this concept, and the BPMN construct is (automatically) annotated with the chosen ESO concept.

The process model to be created in our lab demonstration represents the loan application assessment process in a bank, and is taken from (Dumas, La Rosa, Mendling and Reijers, 2013). By using an existing specification, we avoid bias towards our method. The process starts when the loan officer receives a loan application from one of the bank's customers. This loan application is approved if it passes two checks: the first check is the applicant's loan risk assessment, which is done automatically by the system after a credit history check of the customer is performed by a financial officer. The second check is a property appraisal check performed by the property appraiser. After both checks are completed, the loan officer assesses the customer's eligibility. If the customer is found to be not eligible, the application is rejected. Otherwise, the loan officer starts preparing the acceptance pack. He also checks whether the applicant requested a home insurance quote. If he did, both the acceptance pack and the home insurance quote are sent to the applicant. If the insurance was not requested, only the acceptance pack is sent. The process finally continues with the verification of the repayment agreement

Figure 5 represents the BPMN model of the loan application process. Constructs that are surrounded by a thick red square are annotated with ESO concepts. At this stage, a full visual BPMN modelling tool that implements the suggestion generation algorithms is still under development. However, the suggestion algorithm itself, including all four suggestion mechanisms, is implemented as a stand-alone engine, and available from the repository (i.e., the BPMNSuggestionEngine class). In the remainder of this section the suggestion generation algorithms described in section 3 is demonstrated for the different illustrative scenarios.

Adding Branch Pool: the modeller selects the pool construct to be created. Based on the construct matching mechanism all ESO concepts corresponding to UFO-U object type are given a relevance score of 1 for this matching mechanism. Among those concepts the modeller can find Branch which is classified as UFO-U base type. If there is already a "Customer" pool, based on the construct neighbourhood matching mechanism, this case corresponds with rule 1. As the already existing pool is the Customer pool, the mechanism looks for ESO concepts related to the Customer concept through material relationships. There is only one concept satisfying this requirement: Branch. As a result, the Branch concept is listed in the beginning of the suggestion list, as it scored for both the construct and neighbourhood matching mechanisms (and no other concept scored equal or higher). Note that in this scenario, string and synonym matching cannot contribute to the overall relevance score yet, as the modeller did not (yet) type any label.

Adding Message flow "loan Application": According to the construct matching mechanism, message flow corresponds to the relator universal. Therefore, all ESO concepts corresponding to the UFO-U relator universal will be selected. Those concepts are: Channel, loan, mortgage loan, current mortgage loan, future mortgage loan, invoice, liability, payment. For the element neighbourhood-based matching technique this situation resolves under rule 2.

[6]http://dip.semanticweb.org/documents/D10.2eBankingCaseStudyDesignandSpecificationofApplicationfinal.pdf

In our enterprise-specific ontology, all relator universals are mediating the same two concepts Branch and Customer. Therefore, the results delivered by this suggestion generation technique are the same as the results delivered by construct matching. In this case, the previously mentioned suggestions all have equal overall score, and can thus not be prioritized. We therefore present them alphabetically. The modeller may select a concept from the list (i.e., “loan”), or, in case the list is too long, start typing any desired label (e.g., “loan” or “credit”). This triggers the string- and synonym-based matching mechanisms, both of which prioritize the concept Loan, which consequently appears on top of the suggestion list, and is selected by the modeller to annotate the loan application message flow.

Adding Reject Application Task: The modeller selects the BPMN task construct, and subsequently the construct matching mechanism assigns a high relevance score to all ESO concepts that correspond to UFO-U quality and relator universals as suggestions for the task noun. A list of relator universals is mentioned in the previous example; a list of quality universals is very exhaustive and is thus not mentioned here. The second mechanism, element neighbourhood based matching, applies rule 3: the task at hand is located in the Branch pool, so this matching mechanism suggests all the ESO concepts corresponding to UFO-U relator universals related to Branch concept in the ESO, and UFO-U object types mediated by those relators (all with relevance score 1). The Loan concept is a relator universal, and therefore received relevance score of both matching mechanisms; it therefore appears on the top of the suggestions list.

Adding “Home Insurance Quote is Requested” Gateway: In the last scenario, the modeller draws an inclusive decision gateway on the canvas. Based on construct matching mechanism, all quality universals will be assigned a priority score of 1. The element neighbourhood-based mechanism classifies this situation under rule 4, which suggests ESO concepts that were used to annotate a task construct preceding the gateway. In this case it is the “check if home insurance quote is requested” task, which is annotated with the Home Insurance concept. This concept thus receives relevance score 1 for the gateway, and is prioritized in the suggestion list. It perfectly matches our needs.

To start the discussion, we note that the scenarios elaborated here were chosen to illustrate the more complicated cases. As a result, string- and synonym-based matching mechanism are underrepresented. Evidently, when no or few BPMN elements are already on the canvas, neighbourhood-based matching will be unable to sufficiently differentiate between potential suggestions (as in scenario 2), and string- and synonym-matching will become important. Equally, when the modeller has a certain label already in mind, string- and synonym matching will dominate the suggestion list, as the modeller is typing the label he had in mind. Having made this comment, we note that in general, it was possible to derive suggestions based on the construction and element neighbourhood matching mechanisms for all model constructs for which the related concepts existed in the ontology. In fact, as can be seen in the scenarios, construct and neighbourhood based matching complement each other well. The majority of the concepts required by the model, but missing from the ontology were also correctly classified under the assumptions of neighbourhood-based matching mechanism, and would have been assigned

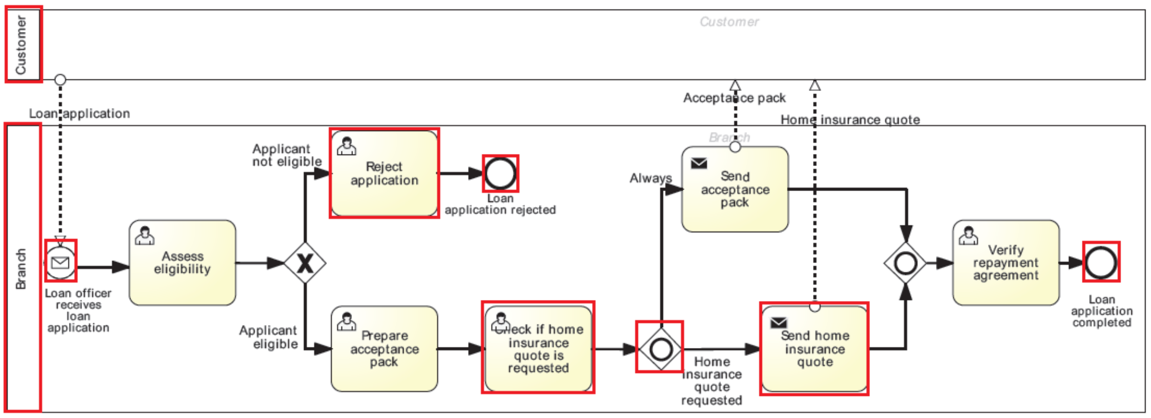
Customer
Customer
Loan application
Acceptance pack
Home insurance quote
Branch
Branch
Applicant not eligible
Reject application
Loan application rejected
Always
Send acceptance pack
Assess eligibility
Loan officer receives loan application
Verify repayment agreement
Loan application completed
Applicant eligible
Prepare acceptance pack
Check if home insurance quote is requested
Home insurance quote requested
Send home insurance quote

Figure 5: BPMN model describing loan application.

a high relevance score if they would have been present in the ontology. However, there was one case where the neighbourhood-based matching mechanism was not very accurate. While creating the last message flow "Home insurance quote", based on the second assumption of the neighbourhood-based matching mechanism and the construct matching mechanism, relator universals must be suggested. But in reality, it was annotated with a quality universal HomeInsurance, instead of a relator. Further fine-tuning of the suggestion generation mechanism should avoid this type of mismatches.

The lab demonstration was used here to demonstrate viability of our method, to detail the different steps and provide a concrete case. It shows that the suggestion generation algorithm indeed provides useful suggestions to the modeller, and allows (automatic) annotations of the model, thereby semantically grounding them and facilitating model integration. It needs to be mentioned that the lab demonstration was done using a single modeller, and that therefore the aforementioned positive indications of using our method cannot be statistically proven. We are currently performing a more elaborate empirical validation, where a group of 30 test users is divided in three different groups: one group is given an ESO and our method, the second group is only given the ESO but without support of our method, and the last group is not given an ESO and thus needs to model without any ontology or method support. The experiment is specifically designed to show the impact of our model on modelling efficiency, consistency in the use of terminology, and the semantic grounding of the resulting models.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we detail a suggestion generation algorithm for BPMN modelling based on an enterprise-specific ontology (ESO), and showed how using our modelling approach allows to automatically annotate the BPMN models with ontology concepts. This work is part of a larger ongoing project, which aims to explore the symbiotic relationship between ontology modelling on one hand, and business modelling on the other hand. The suggestion generation algorithm utilizes four matching mechanisms to derive suggestions from an ESO. Two of them, the string and synonym matching mechanisms, are based on the label of the newly created BPMN element, which is systematically compared with concepts in the ESO. The other two, namely construct matching and neighbourhood-based matching, depend on the type of the BPMN construct and the position (relative to other modelling elements) where it is added. A core ontology, in which the ESO is grounded and with which the modelling language (BPMN) was analysed, proves invaluable for these techniques. Every matching mechanism assigns a relevance score to every ESO concept; the final relevance score is calculated based on the weighted average of scores assigned by every mechanism. This final relevance score forms the basis for prioritizing ESO concepts in the suggestion list, which is offered to the modeller as he is modelling.

Offering the modeller with ESO-based suggestions aids the modeller, facilitates the modelling process, promoted cross-model re-use of ESO concepts, allows model annotation and ultimately, facilitates model integration.

Future work will follow different directions. First, we are currently performing further empirical validation of the benefits of our method. Second, suggestions towards the ontology (e.g., missing concepts) based on the modelling process, and subsequent community-based ontology evolution, needs to be explored. Finally, we also plan to apply the meta-method for other modelling languages (i.e. i*, KAOS), and using other core ontologies.

REFERENCES

Barcellos, M. P. Falbo, R., Dal Moro, R., 2010. A well-founded software measurement ontology. *6th International Conference on Formal Ontology in Information Systems (FOIS 2010)*, (vol. 209, pp. 213-226).

Becker, J., Rosemann, M., Uthmann, C. 2000. Guidelines of business process modeling. In W. Aalst, J. Desel, & A. Oberweis (Eds.), *Business Process Management SE - 3* (Vol. 1806, pp. 30–49). Springer Berlin Heidelberg.

Bera, P., Burton-Jones, A., & Wand, Y., 2011. Guidelines for designing visual ontologies to support knowledge identification. *Mis Quarterly*, *35*(4), pp. 883–908.

Euzenat, J., Shvaiko, P., 2013. *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Fox, M. S., 1992. The TOVE Project: Towards a common-sense model of the enterprise. (F. Belli & F. J. Radermacher, Eds.) *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Berlin, Germany: Springer-Verlag.

Fox, M., Gruninger, M., 1997. On ontologies and enterprise modelling. Enterprise Engineering and Integration. pp 190-200. Springer Berlin Heidelberg.

Francescomarino, C., Tonella, P., 2009. Supporting ontology-based semantic annotation of business processes with automated suggestions. In T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, & R. Ukor (Eds.), *Enterprise, Business-Process and Information Systems Modeling SE - 18* (Vol. 29, pp. 211–223). Springer Berlin Heidelberg.

Gailly, F., Laurier, W., Poels, G., 2008. Positioning and formalizing the REA enterprise ontology. *Journal of Information Systems*, *22*(2), 219.

Guizzardi, G., Wagner, G., 2008. What's in a relationship: An ontological analysis. *Lecture Notes in Computer Science*, *5231*, pp. 83–97.

Guizzardi, G., Wagner, G., 2011. Can BPMN be used for making simulation models? In J. Barjis, T. Eldabi, & A. Gupta (Eds.), *Enterprise and Organizational Modeling and Simulation SE - 8* (Vol. 88, pp. 100–115). Springer Berlin Heidelberg.

Haller, A., Gaaloul, W., Marmolowski, M., 2008. Towards an XPDL compliant process ontology. *Services - Part I, 2008. IEEE Congress on*.

Klein, M., Bernstein, A., 2001. Searching for services on the semantic web using process ontologies. In *Proceedings of the International Semantic Web Working Symposium* (pp. 1–16).

Leopold, H., 2013. *Natural Language in Business Process Models*. Springer Berlin / Heidelberg.

Liao, Y., Lezoche, M., Loures, E., 2013. Semantic enrichment of models to assist knowledge management in a PLM environment. *On the Move toMeaningful Internet Systems*, pp. 267–274.

Lin, Y., Krogstie, J., 2010. Semantic annotation of process models for facilitating process knowledge management. *International Journal of Information System Modeling and Design*, *1*(3), pp. 45–67.

Lin, Y., Strasunskas, D., Hakkarainen, S., Krogstie, J., Solvberg, A., 2006. Semantic annotation framework to manage semantic heterogeneity of process models. (E. Dubois & K. Pohl, Eds.)*Advanced Information Systems Engineering*, *4001*, pp. 433–446.

Miller, G., 1995. WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39–41.

OMG., 2006. Ontology definition metamodel: OMG Adopted Specification (ptc/06-10-11). Object Management Group.

Schlenoff, C., Ivester, R., Knutilla, A., 1998. A robust process ontology for manufacturing systems. In *Proceedings of 2 nd International Conference on Engineering Design and Automation*.

Sonnenberg, C., Huemer, C., Hofreiter, B., Mayrhofer, D., Braccini, A., 2011. The REA-DSL: A domain specific modeling language for business models advanced information systems engineering. In H. Mouratidis & C. Rolland (Eds.), *Lecture Notes in Computer Science 6741* (Vol. LNCS 6741, pp. 252–266). Springer Berlin / Heidelberg.

Uschold, M., King, M., Moralee, S., Zorgios, Y., 1998. The enterprise ontology. *The Knowledge Engineering Review: Special Issue on Putting Ontologies to Use*, *13*(1), pp. 31–89.

Winkler, W., 1990. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association.*