Please cite as follows:

Gailly, Frederik ; Alkhaldi, Nadejda ; Casteleyn, Sven ; Verbeke, Wouter
Recommendation-based Conceptual Modelling and Ontology Evolution Framework (CMOE+),
BUSINESS & INFORMATION SYSTEMS ENGINEERING, 59 (4), pp. 235-250, 2017, ISSN: 2363-7005.

Recommendation-based Conceptual Modelling and Ontology Evolution Framework (CMOE+)

Prof. Dr. Frederik Gailly[a,1]

Ms. Nadejda Alkhaldi[b] ,

Prof. Dr. Sven Casteleyn[c, b]

Prof. Dr. Wouter Verbeke[b]

[a] *Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium*

frederik.gailly@ugent.be

[b] *Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium*

nadejda.alkhaldi@vub.ac.be

wouter.verbeke@vub.ac.be

[c] *Universidad Jaume I, Av. de Vicent Sos Baynat, s/n, 12071 Castellon, Spain*

sven.casteleyn@uji.es

[1] Corresponding Author, tel 00322643477

Recommendation-based Conceptual Modelling and Ontology Evolution Framework (CMOE+)

Abstract. Within an enterprise, various stakeholders create different conceptual models, such as process, data, and requirements models. These models are fundamentally based on similar underlying enterprise (domain) concepts, but they differ in focus, use different modelling languages, take different viewpoints, utilize different terminology, and are used to develop different enterprise artefacts; as such, they typically lack consistency and interoperability. This issue can be solved by enterprise-specific ontologies, which serve as a reference during the conceptual model creation. Using such a shared semantic repository makes conceptual models interoperable and facilitates model integration. The challenge to accomplish this is twofold: on the one hand, an up-to-date enterprise-specific ontology needs to be created and maintained, and on the other hand, different modellers also need to be supported in their use of the enterprise-specific ontology. In this article, we propose to tackle these challenges by means of a recommendation-based conceptual modelling and an ontology evolution framework, and we focus in particular on ontology-based modelling support. To this end, we present our framework for Business Process Modelling Notation (BPMN) as a conceptual modelling language, and focus on how modellers can be assisted during the modelling process and how this impacts the semantic quality of the resulting models. Subsequently, we present a first, large-scale explorative experiment involving 140 business students to evaluate the BPMN instantiation of our framework. The experiments show promising results with regard to incurred overhead, intention of use and model interoperability.

Keywords. *Conceptual modelling, Enterprise ontology, BPMN, Ontology-driven modelling, UFO*

1 Introduction

Conceptual models are used by enterprises to describe formal aspects of the physical and social world for the purpose of communication and understanding (Mylopoulos 1992). As the various stakeholders of an enterprise have different backgrounds and knowledge, they each use different modelling languages in order to achieve their specific goals. This results in conceptual models (e.g. requirements, data, process models) that are not interoperable and are hard to integrate (Hahn 2005; Hofferer 2007; Becker et al. 2009b).

To solve this model interoperability problem, researchers from different fields have proposed using ontologies, albeit in distinctive ways. One research line proposes enterprise ontologies (e.g. Uschold et al. 1998; Geerts & McCarthy 1999), which describes shared concepts and relations across enterprises – to promote model interoperability. Enterprise ontology facilitates the modelling process by suggesting a limited set of enterprise concepts and relationships. However, it also constrains the freedom of the modeller, who is obliged to use generic ontological enterprise elements instead of well-known, conventional terms within

his/her enterprise. Another downside is that the specificities of the particular enterprise and its domain may not be reflected in the generic enterprise ontology.

A second research line uses an ontology that is specifically developed for a particular enterprise, sector or application. This ontology is used to either suggest labels for the model elements (Delfmann 2009; Becker et al. 2009b), annotate the model elements (Born et al. 2007; Thomas et al. 2009), or achieve a combination of both (Francescomarino et al. 2011). In this case, the ontologies describe the concepts, relations and axioms that are typical of and shared within a particular enterprise; they should therefore be considered *enterprise-specific ontologies* (ESOs). The main benefits of this approach are that the ontology can be fine-tuned to the specific enterprise-context and, as opposed to most enterprise ontology approaches, no custom modelling elements or language are imposed. The drawbacks are the lack of guidance during modelling and the additional effort required (as annotations are mostly added after model creation), as well as the fact that the ESO quickly becomes extensive and complex, and therefore difficult to manage, keep up-to-date and use.

In this article, we present a novel, holistic approach to assist conceptual modellers within an enterprise in creating semantically annotated, better interoperable and integrable models by means of an ESO. At the same time, this ESO is maintained and developed in order to reflect the evolving enterprise. Essentially, we propose a generic framework called CMOE+ (Recommendation-based **C**onceptual **M**odelling and an **O**ntology **E**volution Framework) that puts the enterprise's knowledge encoded in the ESO to good use: we use it to recommend relevant concepts and relationships to the modeller which can be used as labels for a model element, and to automatically semantically annotate the models by means of the chosen ESO concepts/relationships. Furthermore, the ESO evolution process is steered by the feedback we collect on the use of modelling suggestions. CMOE+ thus establishes a symbiotic relationship between conceptual modelling, on the one hand, and ESO maintenance and evolution, on the other. With CMOE+, we manage to overcome the drawbacks of both

above-mentioned research lines by combining their advantages. Firstly, we recognize that a well-developed, up-to-date ESO is beneficial for enterprises: apart from contributing to the resolution of interoperability issues, it also serves as a knowledge base incorporating concepts and relations that are used throughout the enterprise. Secondly, we acknowledge that enterprises already have a way of working and that certain workflows, preferred modelling languages and artefacts, or IT tools are already in use. Our framework therefore does not impose new working procedures or a rigid, generic ontology or custom modelling language, but instead is designed to support existing, well-known modelling approaches. Thirdly, we recognize that the ESO will contain a large number of concepts and that, as a consequence, a recommendation mechanism is needed to keep the effort involved under control. We therefore believe that the presented framework incorporates a tangible contribution to the state-of-the-art in the field.

As mentioned, CMOE+ is a generic framework: it defines and implements our modelling method's workflow, along with common functionalities (e.g. recommendation functions, semantic annotation mechanisms, feedback capturing), and it may be instantiated and further specialized to support different concrete modelling languages. In this article, we present one such concrete (partial) instantiation, CMOE+BPMN, which provides recommendation-based modelling support for business process modelling (BPMN). Finally, using an extensive explorative experiment, we evaluate the presented framework, and discuss its impact on the semantic quality of the resulting models, the model interoperability, the time and effort required, their usefulness, and community acceptance.

2 Related work

Existing ontology-based approaches to enhance model interoperability can be classified along two dimensions: (1) approaches that indirectly promote interoperability by means of the modelling language, versus approaches that directly impact on the conceptual model itself

(Hofferer 2007), and (2) approaches that enforce interoperability while creating the model (i.e. avoiding model variations), versus those that create interoperability after the model is created (i.e. managing model variations) (Becker et al. 2009b). These dimensions will be used to review the relevant literature below (see Figure 1).

Within the UEML (Unified Language for Enterprise Modelling) project, the constructs of different conceptual modelling languages are mapped to an intermediate language, which has its origin in the Bunge Wand Weber ontology. Next, these ontological mappings are used to create interoperability between models (Opdahl et al. 2012). The Enterprise ontologies mentioned in the introduction (Uschold et al. 1998; Geerts & McCarthy 1999) are mostly used to develop an enterprise modelling language which is immediately applied during the creation of the model. The work of Becker et al. (2009a), which is based on the ideas of Pfeiffer (2007), uses a domain-specific modelling language to constrain modelling choices, aiming to avoid model variations and promote interoperability.

Approaches that focus directly on the model, as our approach does, use either ontology annotation or matching techniques. For instance, the approach proposed by Born et al. (2007) and Di Francescomarino and Tonella (2009) considers the process model as given and includes an easy-to-use mechanism to annotate these models with elements of an ontology. Another example is the work of Pittke et al. (2013), which focuses on locating inconsistencies within model repositories by identifying synonyms and homonyms by means of matching techniques. As a third example, Becker et al. (2009b) and Delfmann (2009) force the modeller to use naming conventions while s/he adds labels to the model. These naming conventions have their origin in a set of domain terms and phrase structures, and are validated with matching techniques.

What is important to note is that in the process modelling domain, semantically enriched process models are not only used to promote interoperability between process models. They can also be used to automatically analyze business processes (Becker et al. 2010; Fill 2011a;

Fill 2012) or as semantically enriched, machine-readable process specifications for a semantically enhanced process engine (Hepp & Roman 2007; Leutgeb et al. 2007). As a consequence, different authors have proposed languages or frameworks that support adding ontological annotations to process models (Thomas et al. 2009; Fill 2011b) or allow transforming a process model into a semantic business process (Hepp et al. 2005; Abramowicz et al. 2007; Cabral et al. 2009).

CMOE+, the framework described in this article, is classified as an Exaptation in the design science research knowledge contribution framework of Gregor and Hevner (2013), in the sense that known solutions are adapted to a new problem context. With respect to using known solutions, CMOE+ falls in the bottom right classification: during model creation, it (automatically) semantically annotates model elements. CMOE+ additionally addresses the problem of finding the correct ontology concept to annotate with, hereby recognizing the sheer number of concepts typically present in a domain or enterprise ontology. To this end, recommendation mechanisms are proposed to rank ontology elements according to different criteria (see section 3.3) and recommend these to the user during modelling. As such, no restrictions regarding modelling language, structure of models, or use of labels are imposed; instead, the user is guided towards consistent and correct use of terminology within the enterprise ontology. In contrast to related work, where in some cases small-scale validations were performed, we present a large-scale experiment to evaluate various aspects of the presented approach (see section 5). Although this is not the main focus of this article, it is noteworthy that CMOE+ also supports the evolution of the ontology and in fact uses feedback gathered during recommendation- and ontology-assisted modelling to help develop the ontology.

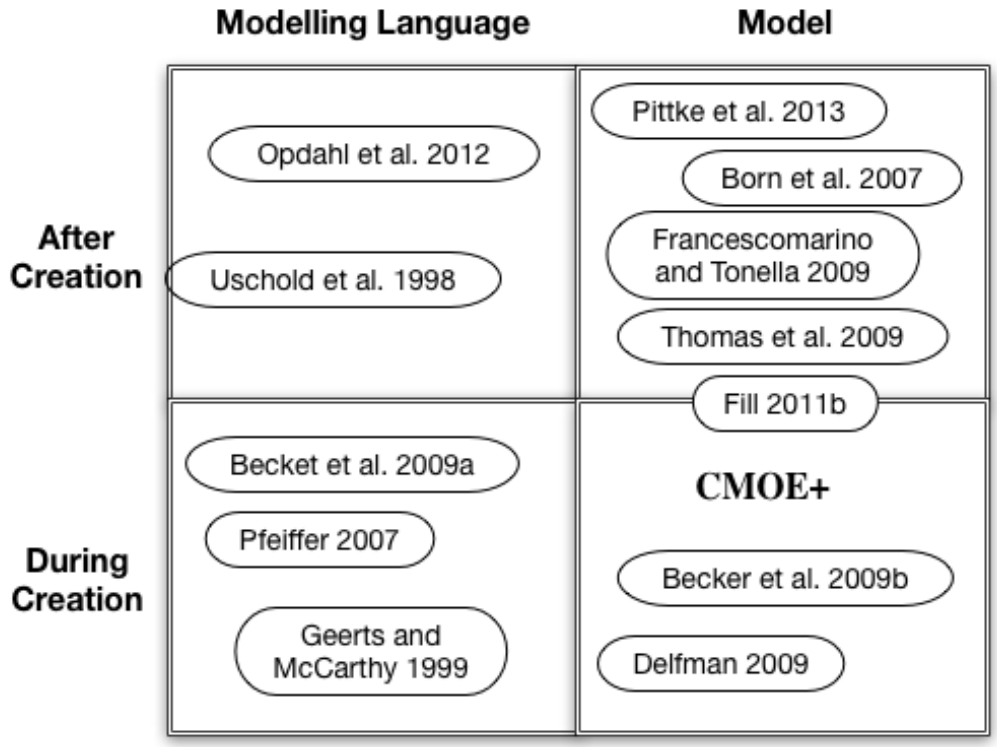

Figure 1: Overview of related research

3 Recommendation-based Conceptual Modelling and Ontology Evolution (CMOE+) framework

The CMOE+ framework was conceived through the Design Science Research Methodology (DSRM) (Hevner et al. 2010), a sound theoretical framework that guides design research and aims at constructing artefacts that solve real-world problems. CMOE+ is one of these artefacts, and is represented in Figure 2. The java implementation of the CMOE+ framework is publicly available (Gailly 2016). It consists of two cycles, the Conceptual Modelling (CM) and Ontology Engineering (OE) cycle, and establishes a symbiotic relationship between these. This paper describes the development and evaluation of the ontology-assisted modelling part of CMOE+; the ontology feedback and evolution part will be the subject of a forthcoming publication. The next subsections give a detailed description of the ontology setup, the ontological analysis of the modelling languages, the ontology storage, the recommendation services, and the model creation phases of the CMOE+.

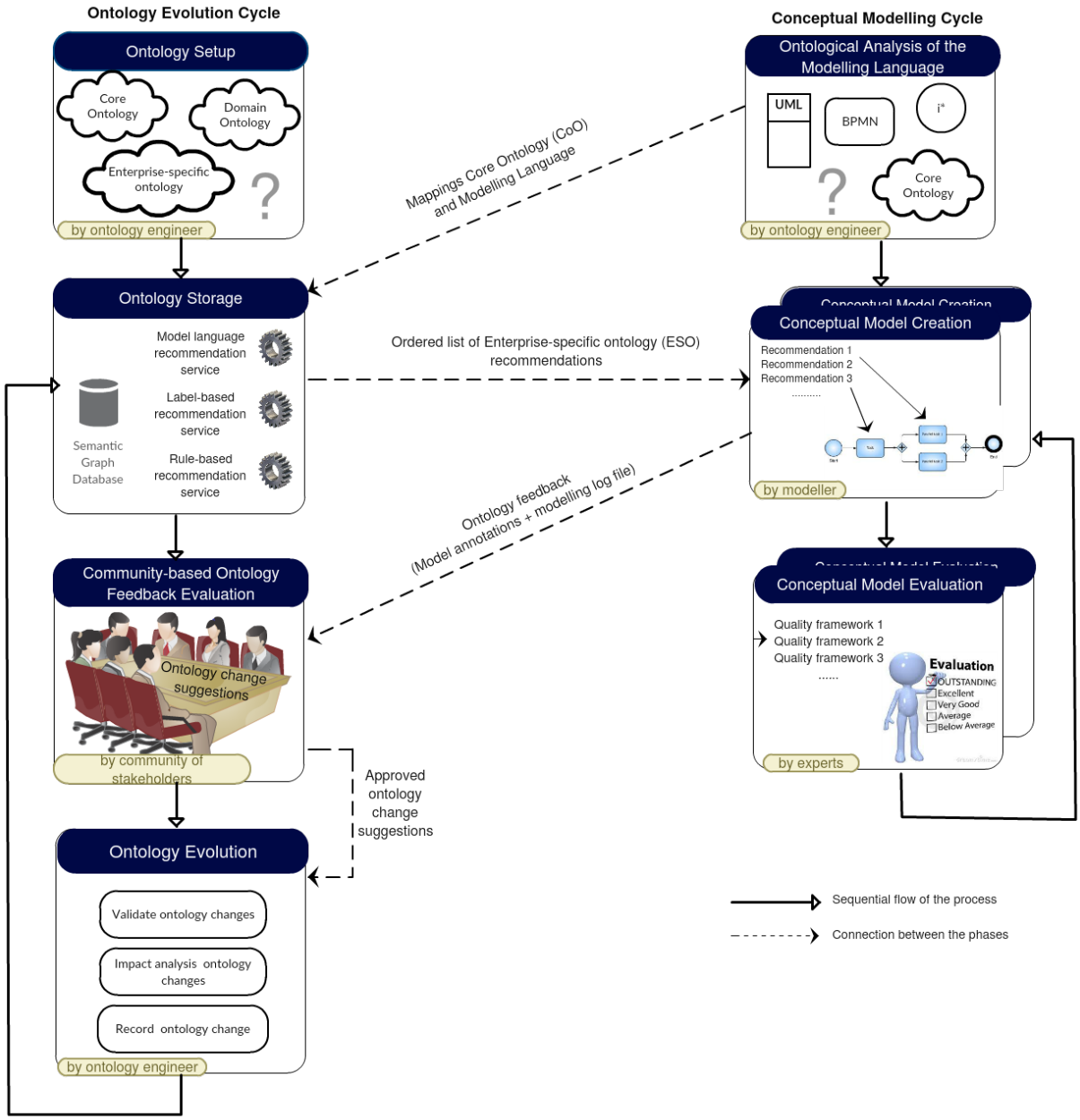

Figure 2: Recommendation-based Conceptual modelling and Ontology Evolution (CMOE+) framework

3.1 Ontology Setup

The OE cycle commences with the *Ontology Setup* phase, in which the enterprise decides which ESO it will take as a starting point. The ESO can be created by means of an existing ontology engineering method (for an overview, see Suárez-Figueroa et al. (2012)) and with available business resources (e.g. glossaries, vocabularies, informal sources such as excel files of use case descriptions) as input. Additionally, the enterprise may start from an existing domain ontology that covers the business domain (e.g. the Resource Event Agent Enterprise

ontology by Geerts and McCarthy (1999) or the Enterprise Ontology by Uschold et al. (1998)) and that is gradually transformed into the ESO. Once developed, the ESO needs to be grounded in a core ontology according to good ontology engineering practice (Guarino 1998). A core ontology describes universally agreed upon, high-level concepts and relations, such as objects, events, or agents (Guarino 1998), and thus provides well-founded semantics, facilitates data integration across different (sub-) domains, and forms the basis for subsequent interoperable application building. CMOE+ does not prescribe a specific core ontology, yet we recommend and provide support for the Unified Foundational Ontology (UFO) (Guizzardi et al. 2015) since re-usable analyses of conceptual modelling languages are available in the literature. Different approaches and tools are available to ground the enterprise-specific ontology in a core ontology. For instance, core ontology patterns can be used to develop or analyze ontologies (Blomqvist 2005; Ruy et al. 2015). Other useful tools for ontology engineers are ONTOCLEAN (Guarino and Welty 2002) and OntoUML (Guizzardi et al. 2015), which can be used to evaluate the grounding of ontology concepts in the core ontology.

3.2 Ontological Analysis of the Conceptual Modelling Language

The first phase of the conceptual modelling cycle is another initialization phase, in which an ontological analysis is performed for the target conceptual modelling language(s) used in the enterprise. Different authors have proposed methodologies and frameworks to achieve this (Evermann and Wand 2005; Harzallah et al. 2012; Guizzardi 2013). The purpose of these methodologies and frameworks is (1) to provide a rigorous definition of the construct of a modelling languages in terms of real-world semantics, (2) to identify inappropriately defined constructs, and (3) to recommend language improvements which reduce a lack of expressivity, ambiguity, and vagueness (Almeida & Guizzardi (2013). In CMOE+, the goal is not to improve the language itself, but to relate the constructs of the conceptual modelling

language to the core ontology selected in the ontology setup phase. These connections can later on be exploited in the conceptual modelling recommendation service (see section 3.4). Over the years, different conceptual modelling languages have been analyzed with, for example, Bunge-Wand-Weber (e.g. UML class diagrams in Opdahl and Henderson-Sellers (2002)) and UFO (e.g. BPMN in Guizzardi and Wagner (2011)). Although the added value of these ontological analyses have generally been accepted, their translation into conceptual modelling practice has been limited. While CMOE+ does not prescribe any particular core ontology, it does currently support ontological analyses using UFO or BPMN (see section 4 for more details) and i* (not reported here).

3.3 Ontology Storage

Efficient ontology storage is essential in order to easily query and update the ontology and ensure efficient recommendation services. Based on our extensive experience with implementing the framework for BPMN and i*, CMOE+ currently supports the Web Ontology Language (OWL)[2] as ontology representation language for various reasons. First of all, it is a generally accepted (W3C) ontology language standard, supported by most ontology engineering tools (e.g. Protégé) and with APIs for various programming languages. In addition, OWL 2.0 supports punning, which is heavily used in our approach (see further in this subsection) (Grau et al. 2008). Finally, OWL offers highly optimized storage media, such as the Stardog semantic graph database[3], which is used as storage medium in CMOE+. This database was selected for ontology storage in CMOE+ because of its support for OWL 2.0, excellent access and querying performance, and support for Java, which is also used by our Eclipse-based modelling tools. Another advantage of Stardog is that it makes CMOE+ ready for a future production-level implementation, as it is specifically optimized to handle huge, highly interconnected datasets.

[2] https://www.w3.org/TR/owl2-overview/
[3] http://www.stardog.com

The Stardog Database consists of different interconnected OWL ontology files. Panel A of Figure 3 gives an overview of the different ontology files and their relationships, while panel B further explains the different ontologies by means of some examples:

- The Core Ontology (CoO) file contains the concepts and relations of the core ontology as OWL classes and OWL object properties, respectively. Currently, our framework only contains a CoO file for the Unified Foundational Ontology. An UFO ConceptType is an example of a CoO concept which can be included in the CoO file.
- The Modelling Language Ontology (MLO) file is a formalization in OWL of the meta-model of the used conceptual modelling languages. It stores the constructs of the language as OWL classes and the properties of the constructs as OWL object properties. The OWL class Pool is an example of a BPMN construct that can be incorporated into the MLO file.
- The CoO-MLO file captures the outcome of the ontological analysis of the modelling languages (see section 3.2), each in a separate OWL ontology file. The mappings between MLO elements and CoO elements are formalized by OWL equivalence relationships. For instance, an OWL equivalence relationship exists between the CoO ObjectType and the MLO Pool.
- The Enterprise-Specific Ontology (ESO) file describes the concepts and relations of the enterprise-specific ontology as OWL classes and object properties, and the hierarchy relationships in the ESO that use OWL specializations relationships. For instance, the ESO contains a Customer OWL class and a Person OWL class, both of which are ESO concepts; furthermore, the Customer OWL class is an OWL (to be precise, RDFS) subclass of the OWL Person class. Additionally, the relationship between the concepts and relationships of the ESO and the CoO is incorporated by means of the OWL punning mechanism, which allows us to define an OWL element as both a class and an individual. Consequently, the concepts and relationships of the ESO are also defined as OWL individuals of the CoO classes and assertions of CoO object properties, respectively. As such, OWL punning allows us to capture the mappings between CoO and ESO by means of instance relationships, which is essential to be able to fully exploit OWL's reasoning capabilities (see section 3.4). Panel B of figure 3

illustrates this by indicating that the ESO Concept is both a class (circle with full line) and an individual (circle with dashed line).

- The Model Ontology (MoO) file is created during the model creation phase (see section 3.5). For every modelling language construct that the modeller adds to his/her conceptual model, an OWL individual is created, whose type is the corresponding element of the MLO. In our example, the Pool Element with the label Customer is an instantiation of the Pool construct captured in the MLO file. In order to also support adding annotations, the MoO file imports the SemAnnO file, which defines the semantic annotation OWL object property that is used to add annotations to the OWL individuals of the MoO file. A similar approach for annotating model elements is applied by (Thomas et al. 2009). This annotation approach was chosen because the rule-based recommendation service requires that the annotations are taken into account during the reasoning process.
- The RulesO file contains Semantic Web Rule Language (SWRL) rules that are used by the Rule-based Recommendation Service to infer new knowledge based on the assertions that are available in the ESO and the MoO. More specifically, the rules may imply semantic annotations through the concepts and relations of the ESO, CoO and the MLO (see section 3.4).

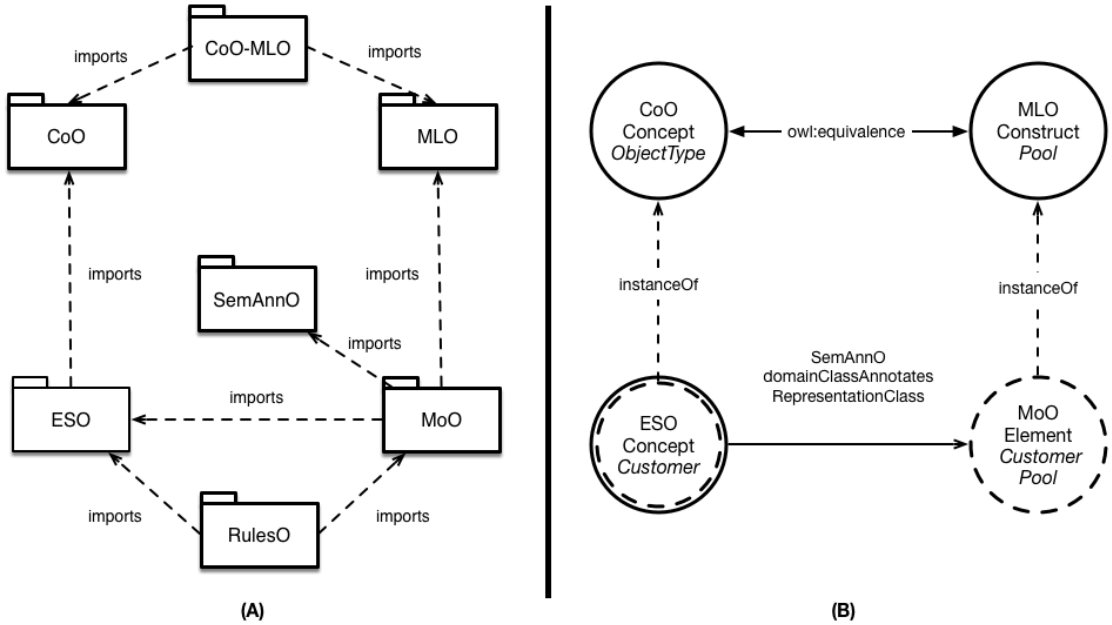

Figure 3: Ontologies CMOE+ framework

3.4 Recommendation Services

Based on the above-mentioned stored ontologies, the recommendation services determine what ESO concepts are suggested to the modeller. For each ESO concept, each recommendation service calculates a recommendation score between 0 and 1, with respect to a modelling element added by the modeller. The final relevance score is a weighted average of all individual recommendation scores, creating a (weak) ranking for suggested ESO concepts (see section 3.5). Consequently, ESO concepts are ordered according to relevance, which is essential to help modellers find appropriate concepts quickly, as the ESO rapidly becomes large and complex. CMOE+ supports three recommendation services:

1. The *model language recommendation service* deduces recommendations based on an ontological analysis of the conceptual modelling language: given a modelling language construct, its associated CoO concepts are derived using ontological analysis mapping and then compared with ESO groundings in CoO concepts. The pseudo code is given in Listing 1. First, a working ontology is considered, merging a selection of ontologies that are available in the framework (line 2). Next, the ontology reasoner is used to extend the ontology with assertions. This is accomplished with both the classification mechanism and realization mechanism of the reasoner (line 3). Here, the added ontology assertions have their origin in the equivalence relations that are defined in the CoO-MLO file, and will result in classifying some of the ESO concepts as individuals of the MLO constructs. After this, the SPARQL query service of the reasoner is used to create a collection of ESO concepts that belong to the type of the modelling language construct that is given as input (line 4 and 5). The FOR EACH block starting in line 6 is a consequence of the punning mechanism. It uses the SPARQL query service of the reasoner to add the subclasses of the existing ESO concepts candidates (lines 7 – 9). Finally, the IF-ELSE block of Line 11 checks whether the ESO concept that is given as input of the algorithm is a member of the created ESO candidates set. If this is the case, the algorithm returns the (individual) recommendation score 1; if not, 0 is returned.

```
Algorithm: Model Language Recommendation Service
Input: ESOConcept, MLconstruct, CoO, MLO, CoO-MLO, ESO
Output: 0 or 1
1  begin
2      ontology ← CoO ∪ MLO ∪ CoO-MLO ∪ ESO;
3      ontology ← Reasoner.reason(ontology);
4      query1 ← SELECT ?x WHERE {?x rdf:type :MLconstruct };
5      ESOcandidates ← Reasoner.query(ontology, query1);
6      foreach element ∈ ESOcandidates do
7          query2 ← SELECT ?y WHERE {?y rdfs:subClassOf :element};
8          subClasses ← Reasoner.query(ontology, query2);
9          ESOcandidates ← ESOcandidates ∪ subClasses;
10     end
11     if ESOConcept ∈ ESOcandidates then
12         return 1;
13     else
14         return 0;
15     end
16 end
```

Listing 1: Pseudo-code Model language recommendation service

It is important to note that in Listing 1, for the sake of simplicity and clarity, we describe the recommendation service that calculates the relevance score for *one* particular ESO concept. In our implementation, such a relevance score is calculated for all ESO concepts, hereby caching static intermediary results (e.g. ESOcandidates) for efficiency.

2. The *label-based recommendation service* uses the ESO and natural language processing techniques (i.e. string and synonym matching) to give a relevance score to an ESO concept based on lexical distance of the concept name (and all its synonyms) and the label that is entered by the modeller. Listing 2 presents the pseudo code. In Line 3, the string matching score is calculated using Jaro-Winkler distance (Winkler 1990) between the label that is entered by the modeller and the label of the ESO concept. Line 4 of the algorithm creates a collection of synonyms for the label of the ESO concept using WordNet (Miller 1995). This collection is used by the FOR EACH block (line 5), which calculates the string matching score between the entered label and every synonym from the collection. The FOR EACH block only remembers the highest matching score. Finally, line 8 returns this stored matching score, which is the (individual) recommendation score.

```
Algorithm: Label-based Recommendation Service
Input: ESOconcept, enteredLabel
Output: score
1 begin
2     conceptLabel ← getLabelConcept(ESOconcept);
3     score ← getStringMatchScore(conceptLabel, enteredLabel);
4     synonyms ← getWordNetSynonyms(conceptLabel);
5     foreach synonym ∈ synonyms do
6         score ←
              max(score, getStringMatchScore(synonym, enteredLabel))
7     end
8     return score
9 end
```

Listing 2: Pseudo-code label-based recommendation service

3 The *rule-based recommendation service* uses the rules specified in RulesO to identify suggestions for labels of modelling element added by the modeller. Listing 3 presents the pseudo code. The algorithm starts with creating a new modelling element (see Line 2) which corresponds to the model element that is currently selected by the modeller and which is not yet annotated. To ensure that the recommendation service takes this element into account, the element is added to an updated version of MoO (i.e. MoO'). Next, similar to the model language recommendation service, the algorithm assembles a new working ontology, which is extended with assertions by the reasoner (see Line 4 and 5). Compared to the model language recommendation service, the rule-based recommendation service also uses the RulesO and MoO' as input, which are used by the rules reasoning service of the reasoner to add new suggestions (in the form of asserted semantic annotations) for the currently selected model element. After reasoning, the algorithm creates a collection which contains all ESO concepts for which the reasoner identified a potential semantic annotation for the new element. If the ESO concept that is given as input of the algorithm is an element of this collection, the algorithm returns 1 as individual recommendation score; if not, 0 is returned.

```
Algorithm: Rule-based Recommendation Service
Input: ESOconcept, MLconstruct, CoO, MLO, CoO-MLO, ESO, MoO,
       RulesO, SemAnnO
Output: score
1  begin
2      newElement ← createElement(MLconstruct);
3      MoO′ ← add(MoO, newElement);
4      ontology ←
           CoO ∪ MLO ∪ ESO ∪ MoO′ ∪ CoO-MLO ∪ RulesO ∪ SemAnnO;
5      ontology ← Reasoner.reason(ontology);
6      query ← SELECT ?x WHERE
           {:hasSemAnn rdfs:domain :newElement rdfs:range ?x};
7      result ← Reasoner.query(ontology, query);
8      if ESOconcept ∈ result then
9          return 1;
10     else
11         return 0;
12     end
13 end
```

Listing 3: Pseudo-code rule-based recommendation service

3.5 The Conceptual Model Creation Phase

In the *Conceptual model creation phase* (CM cycle), the modeller is presented with an ordered list of ESO recommendations, based on the selected modelling language construct and the label entered. The (weakly) ordered list is calculated through a (configurable) weighted average of individual recommendation service scores, which determines the order in which the ESO concepts are presented to the modeller. The modeller is free to accept or discard a recommendation. If s/he accepts a recommendation, the selected model element is automatically annotated with the corresponding ontology concept, and the label of the modelling construct that is added is updated with the name of the selected ESO recommendation. CMOE+ currently supports semantic annotations using OWL. In line with Thomas et al. (2009), the ontology annotation is stored in the MoO by adding an assertion of the semantic annotation object property between the MoO OWL individual and the ESO OWL individual.

Additionally, during modelling and while the process of either adopting or discarding recommendations, feedback is gathered and stored in a log file. This log is stored in the mxml

format which means that it can be processed by the ProM process mining tool[4]. The events that are stored in the log are (1) the generation of recommendations for the label entered, (2) acceptance of a recommendation by annotating the model, and (3) deletion of model annotation.

4 Recommendation-based Business Process Modelling (CMOE+BPMN)

To demonstrate that the CMOE+ framework is a feasible, adequate and efficient solution for the presented problem, it was instantiated for process modelling by means of BPMN. Consequently, we will now move on to describe the CMOE+ recommendation-based business process modelling implementation (i.e. CMOE+BPMN) that uses, specializes and extends CMOE+'s generic functionality. The CMOE+BPMN implementation is an Eclipse plugin which can be downloaded from GitHub[5] and is shown in Appendix D. By means of the eclipse plug-in extension point mechanism, the CMOE+BPMN plug-in extends the Eclipse BPMN2 modeller[6] with two views and a preference page. BPMN2 Modeller is a graphical modelling tool which is built using Eclipse Graphiti in combination with the BPMN 2.0 EMF meta-model. Graphiti is an Eclipse-based graphics framework that enables the rapid development of diagram editors starting from an EMF meta-model. The implementation of the ontology storage and the recommendation services are described in more detail below.

Ontology Storage

The ontologies used for CMOE+BPMN, along with some ontologies that will be applied in our case study (see section 5), are the following:

- The Unified Foundational Ontology (UFO) was selected as a core ontology (i.e. CoO). UFO has different layers, of which only those elements are selected which are relevant in the

[4] http://www.promtools.org, last accessed 5 August 2016
[5] https://github.com/fgailly/CMOEplusBPMN, last accessed 5 August 2016
[6] http://www.eclipse.org/bpmn2-modeler/, last accessed 5 August 2016

context of process modelling for this instantiation of CMOE+. A short description of UFO can be found in Appendix A; for a full explanation, we refer to Guizzardi et al. (2015). The OWL formalization of UFO is available online[7].

- In the demonstration, an existing OWL ontology from the financial domain is selected as enterprise-specific ontology (i.e. ESO). The ESO concepts are formalized as both OWL classes and OWL individuals, as outlined in section 3.3. Throughout this paper, ESO concepts are denoted in *italics*. The mappings between ESO concepts and UFO are presented in Appendix B, and were obtained using the description of the ESO concepts and their intent. For example, ESO *ProductRateApplication* is defined as applied interest rate. This implies that *ProductRateApplication* is a quality of object type *Product*. An ESO *Loan* is intended to relate a *Customer* to the *Branch* s/he took a loan from. Therefore, *Loan* is an instance of the UFO Relator universal relating *Customer* and *Branch*. ESO *LoanApplicationAccepted* is an event representing the acceptance of loan application, thus instantiating an Event type in UFO. The OWL formalization of the bank ontology is available online[8].
- The used BPMN ontology (i.e. MLO) is an OWL translation of the meta-model shown in Figure 4, and is based on the original OMG BPMN standard (OMG 2006). In this paper, we extend OMG meta-model based on the observation that different authors advise BPMN modellers to follow the pattern "verb noun" when they specify the name of a task (Delfmann 2009). The OWL formalization of the BPMN meta-model is available online[9].
- The mappings between UFO and BPMN (i.e. CoO-MLO) are based on the ontological analysis provided by (Guizzardi & Wagner 2011). Table 1 represents the mappings between the constructs of the BPMN meta-model and UFO. Important to notice is that the BPMN Event and the Activity construct are both mapped to an UFO Event type. Moreover data objects and Message flow objects are mapped to Relators (e.g. contracts, invoices), and Base

[7] http://www.mis.ugent.be/ontologies/ufo.owl, last accessed 5 August 2016
[8] http://www.mis.ugent.be/ontologies/bank.owl, last accessed 5 August 2016
[9] http://www.mis.ugent.be/ontologies/bpmn.owl, last accessed 5 August 2016

types (e.g. database, technical documentation of software). The OWL formalization of the mappings is available online[10].

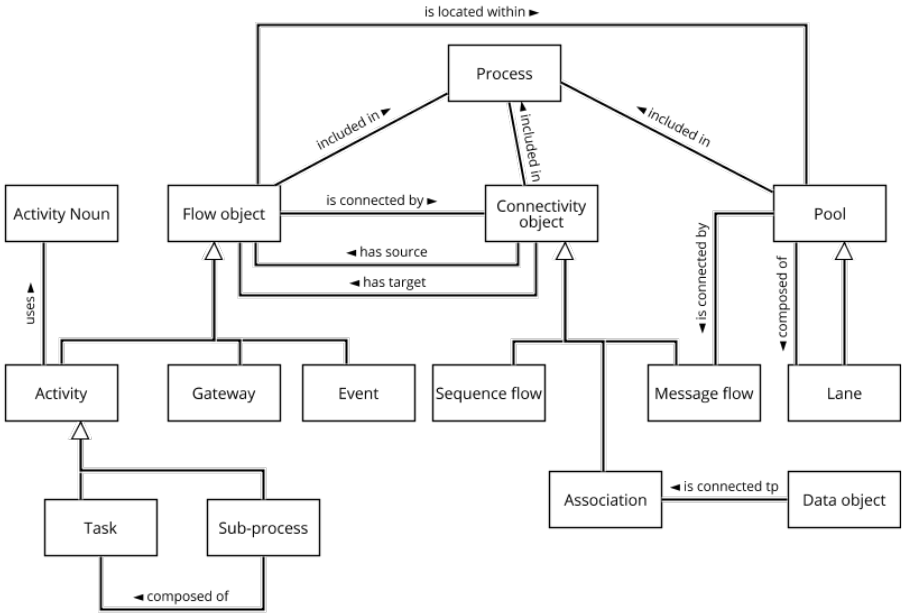

Figure 4: BPMN meta-model

Table 1: Correspondence between BPMN and UFO

BPMN construct	**UFO**	**BPMN construct**	**UFO**
Pool	ObjectType	Event	EventType
Lane	ObjectType	MessageFlow	RelatorUniversal or ObjectType or QualityUniversal
Activity	EventType	Association	MaterialRelationshipType or FormalRelationship_Type
Data object	RelatorUniversal or ObjectType or QualityUniversal		

Recommendation Services

The recommendation services are used by the BPMN editor to arrange the ESO concepts in the ontology property view (see figure 5), which is implemented following the Model-View-Controller pattern. The controller of the ontology recommendation view updates the associated view every time the modeller selects a model element on the canvas. The

[10] http://www.mis.ugent.be/ontologies/bpmn_ufo.owl, last accessed 5 August 2016

CMOE+BPMN tool contains a second view, which is used to give more detailed information about the selected ontology recommendation. The controller of the ontology property view updates the associated view when the modeller selects an ontology recommendation. CMOE+BPMN uses the OWL API[11] to implement the different recommendation services, and the HermiT reasoner (Glimm et al. 2014), included in the OWL API, is used for querying and reasoning. The label-based recommendation service uses CMOE+'s support for the Jaro-Winkler distance (Winkler 1990) to compare Strings and WordNet (Miller 1995) to determine synonyms (see Listing 2). In some cases (i.e. for BPMN tasks, sub-processes, events, and conditional gateways), the label is pre-processed. For this purpose, the Stanford Parser[12] is applied to tokenize the labels.

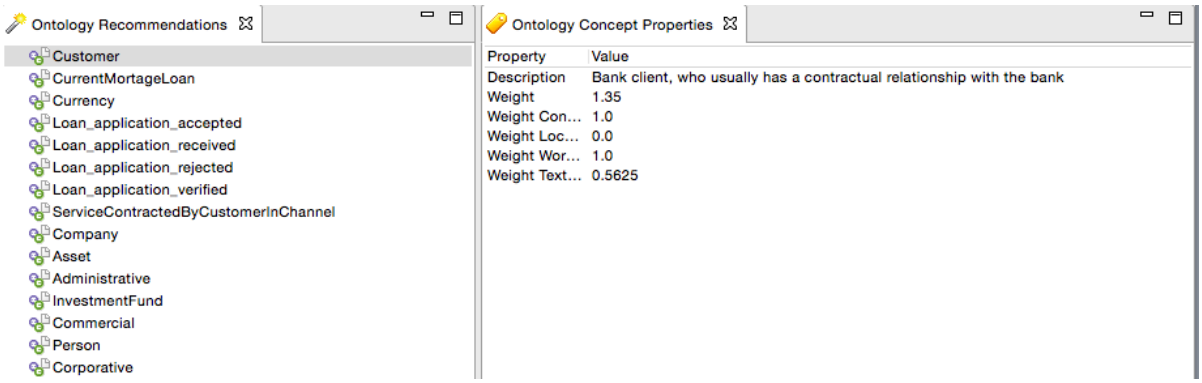

Figure 5: Ontology Recommendation view (Left) and Ontology Concept Properties view (Right)

Using the rule-based recommendation mechanism, BPMN-specific recommendation rules (i.e. RulesO) were added in CMOE+BPMN. The rules that were used in the experiment (see section 5) are listed in Table 2; a full specification can be found online[13]. In future research, we plan to investigate in more detail which kind of rules may be useful to add to this recommendation service.

5 Evaluation of CMOE+BPMN

[11] http://owlapi.sourceforge.net, last accessed 5 August 2016
[12] http://nlp.stanford.edu/software/lex-parser.shtml, last accessed 5 August 2016
[13] http://www.mis.ugent.be/ontologies/cme_bpmn_rules.owl, last accessed 5 August 2016

CMOE+BPMN aims to promote label consistency and facilitate model annotations, while ideally avoiding significant overhead in modelling time and perceived effort. Annotating modelling elements with ontology (ESO) concepts then results in more interoperable models, as previously shown in literature (Born et al. 2007; Di Francescomarino & Tonella 2009; Thomas et al. 2009). This section presents an explorative experiment to empirically validate CMOE+BPMN using Moody's Method Evaluation Model (MEM) (Moody 2003).

Table 2: SWRL rules used by the rules-based recommendation service

BPMN:Pool(?x) ^ BPMN:Pool(?y) ^ SemAnn(?x,?o) ^ UFO:mediates(?r,?o) ^ UFO:mediates (?r,?p) ^ ➔ SemAnn(?y,?p)
This rule indicates that when the modeller creates a pool construct, the UFO object types, which are related to UFO object types that have previously been used to semantically annotate another pool in the model, will be suggested by the rule recommendation service.
BPMN:Pool(?x) ^ BPMN:Lane(?y) ^SemAnn(?x,?o) ^ UFO:mediates(?r,?o) ^ UFO:mediates (?r,?p) ^BPMN:hasLane(?x,?y) ➔ SemAnn(?y,?p).
This rule indicates that when the modeller creates a lane construct within a pool, the suggestions (relevance score 1) are UFO object types that are related by a material relationship with the ontology annotation of the pool.
BPMN:MessageFlow(?x) ^ BPMN: Pool(?y) ^ BPMN:Activity(?z) BPMN:connects(?x, ?y) ^BPMN:connects(?x, ?z) ^ SemAnn(?x,?o) ^ SemAnn(?z,?p) UFO:Relator(?r) ^ UFO:mediates(?r,?o) ^ UFO:mediates (?r,?p) ➔ SemAnn(?x,?r).
When a message construct is created that results in the transmission of a message between a activity of a pool and another pool, the suggestions are UFO relators mediating material relations that connect objects that in turn annotate the noun of the task and the ontology annotation of the pool, respectively.

5.1 Experimental design

Using an identical case description (see Appendix C), modellers were asked to create a BPMN model. Three different treatments were applied: treatment 1 assists modellers with

CMOE+BPMN as described in section 4.3; treatment 2 provides modellers an alphabetically ordered list of ESO concepts, without relevance ordering, so that the modeller needs to find relevant ESO concepts him/herself (see Appendix D[14]); treatment 3, as a baseline, does not provide any modelling support (i.e. regular BPMN modelling). Where relevant (treatment 1 and 2), the modeller was asked to annotate the modelling element with ESO concepts. The BPMN modelling tool described in Section 4 was used to conduct the experiments. An additional view was developed for treatment 2 to support only alphabetical ordering of ESO concepts (without recommendations), and for treatment 3 the recommendations view was disabled.

The participants of our experiment were 140 university students at the master level, who were acquainted with BPMN because they took a mandatory Business Process Management course. The subjects were distributed randomly across the three treatments: 47 for treatments 1 and 2, and 46 for treatment 3. Every group was given a tutorial explaining the tool and the required actions during the experiment.

5.2 Experiment Measures

In Moody's Method Evaluation Model (MEM), the impact of using the method on performance, user perception and intention of use is measured, thus assessing the acceptance of future practitioners. Applying MEM to CMOE+BPMN resulted in six variables to be observed during the experiment: semantic quality, interoperability, time, perceived ease of use, perceived usefulness, and intention of use. These dependent variables were operationalized in the Cheetah experimental platform (Pinggera et al. 2010), which makes it possible to collect answers for the pre- and post-survey (see Appendix E and F), collect the created models and record the time spent on each task.

[14] All appendices are available online at https://github.com/fgailly/CMOEplusBPMN, last accessed 5 August 2016

The first variable, semantic Quality (SQ), was measured by verifying validity (i.e. is every statement in the model correct with respect to the case description?) and completeness (i.e. does the set of all statements completely cover the case?) (Lindland et al. 1994). To measure validity and completeness, for every model, the number of invalid and missing statements were counted, respectively, in comparison with a reference model created by a team of three BPMN modelling experts (Appendix G).

The second observed variable was interoperability (I). CMOE+BPMN was expected to enhance interoperability across models (1) by providing ESO-based recommendations and automatically annotating BPMN labels, which promotes the reuse of ESO concepts in model element labels, and (2) by consistently recommending the same ESO concept for similar labels, which promotes model consistency and thus interoperability. The degree of model interoperability was measured by counting the number of annotations in every model (treatment 1 and 2). In addition, to verify consistency, the variation in labelling of modelling elements with the same underlying meaning was assessed by examining the distribution of labels of such elements across different models of one treatment (all treatments).

The third observed variable was time spent for creating the model (T). The aim was to determine if time overhead was incurred by turning to vocabulary support or not. In our experiment, time was measured by the Cheetah platform, starting when the participants began model creation, and stopping when the final model was uploaded.

All other variables were measured using a post-experiment survey (see appendix F). The perceived ease of use (PEOU) and perceived usefulness (PU) of the method were measured by adapting the generally accepted measurement scales of Davis (Davis 1989), with three different questions. Intention of use (IU) was measured by means of two questions in the post-experiment survey. All answers were provided on a Likert scale from one (strongly disagree) to five (strongly agree).

5.3 Experimental results

Before analysing the results, we performed a pre-selection of models based on syntactic quality: models with more than two mistakes against the BPMN specification were discarded to eliminate qualitatively insufficient models[15] and reflect a real-life setting in which syntactically incorrect models are improved before acceptance or discarded.

For the retained models, we analysed the results for the six variables prescribed by MEM. Statistical significance was tested using the Mann-Whitney test for SQ, PEOU, PU, IU as they are ordinal variables, and for T and I as they are not normally distributed continues variables. Normality of the distribution was tested with Kolmogorov-Smirnov and Shapiro-Wilk tests. Statistical significance of label distribution among models was evaluated using chi-square analysis to determine the likeliness of the observed label distribution occurring by chance, independently of the treatment. For all test, the results were considered statistically significant if the p-value was < 0.05. In all tables, only statistical significant results are explicitly denoted; all other differences were not statistically significant.

Table 3 shows the results of the **Semantic Quality** (SQ) evaluation. We found no statistically significant difference between the treatments for validity, and thus conclude that ontology support does not decrease validity. For semantic completeness, we found no statistically significant difference between Treatment 1 and Treatment 2, yet both performed significantly worse than Treatment 3. Observation during the experiments indicated that participants from Treatment 1 faced some technical issues with the tool, which could have caused them to concentrate more on the functioning of the tool itself, rather than producing a complete model. Furthermore, the tutorial participants received was focused on vocabulary support, which may have caused them to perceive the experiment as a test in vocabulary

[15] Note that the reference model corresponding to the case study only contains 14 BPMN constructs; more than two errors is thus high and indicates poor model quality.

usage, relaxing their focus on the modelling and model completeness. These possible influences should be eliminated in follow-up experiments.

The results for **Interoperability** are shown in Table 3 (number of annotations) and Tables 4 and 5 (naming variation). Considering average and median percentages of annotated modelling elements per treatment (Table 3), roughly 70% of BPMN elements were annotated with an ESO concept. Overall, CMOE+BPMN (Treatment 1) performs slightly better than the other two, but the observed differences were not statistically significant. The number of fully annotated models for Treatment 1, however, is more than twice the number for the other treatments. We can therefore conclude that, if given the possibility, modellers annotate a large portion of their modelling elements, thus increasing model interoperability. Furthermore, customized recommendations, as provided by CMOE+BPMN, increase the number of fully annotated models.

Table 3: Results of semantic quality evaluation model annotations

		T 1	**T 2**	**T 3**	**Statistical analysis**
Total number of models		47	47	46	
Number of models evaluated		24	31	20	
Semantic Quality	Number of models without validity issues	18 (75 %)	24 (77.42 %)	18 (90 %)	
	Number of models with 1 invalid statement	4 (16.7 %)	7 (22.58 %)	1 (5 %)	
	Number of models with 2 invalid statements	2 (8.3 %)	0	1 (5 %)	
	Number of complete models	1 (4.2%)	11 (36%)	7 (35%)	T1⇔T3: significant T2⇔T3: significant
	Number of models with 1 missing statement	12 (50%)	10 (32%)	6 (30%)	T1⇔T3: significant T2⇔T3: significant
	Number of models with 2 or more missing statements	11 (45.8%)	10 (32%)	7 (35%)	T1⇔T3: significant T2⇔T3: significant
Model	Average number of annotations	70.38%	66.98%		

	Median of annotation	78.57%	71.43%		
	Fully annotated models	5 models (20.83%)	3 models (9,68%)		
	Models with no annotation	2 models (8.33%)	1 model (3.23%)		

Considering the consistency of labels, Table 4 presents the results of naming distribution across models for elements referring to a *customer* (i.e. a single BPMN pool), whereas Table 5 shows the results for three different modelling elements featuring *loan application* (i.e. a start event *(loan application received*) and two different end events (*loan application rejected; loan application accepted*)). Multiple instances of the same event, or an event and a task with the same meaning were not counted. In the first column, we also denote the theoretical maximum number of uses, not counting any models that lack an individual modelling element. We can observe that for "*customer*" (Table 4) and "*loan application*" (Table 5), Treatments 1 and 2 performed statistically significantly better compared to Treatment 3: the label corresponding to an ESO concept was used in around 85% of the cases, while results were more dispersed without vocabulary support. With vocabulary support (i.e. Treatments 1 and 2), modellers thus consistently opt for the correct underlying ESO concept, which more clearly corresponds with the underlying business domain and increases the consistent use of labels. Overall, we can conclude that vocabulary support improves interoperability.

Table 4: Naming for BPMN elements with underlying meaning "*customer*". Columns are modeller-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning "*customer*"

	Customer (ESO concept)	Client	Person	Applicant
T 1	14/18 (77.78%)	0	2/18 (11.11%)	2/18 (11.11%)
T 2	23/27 (85.19%)	0/27	4/27 (14.81%)	0/27
T 3	9/18 (50%)	9/18 (50%)	0/18	0/18

Table 5: Naming for BPMN elements with underlying meaning "*loan application*". Columns are modeller-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning "*load application*"

	Loan application (ESO concept)	Loan	Application	Request
T 1	57/62 (91.95%)	1/62 (1.61%)	2/62 (3.22%)	2/62 (3.22%)
T 2	75/85 (88.23%)	3 (3.53%)	3/85 (3.53%)	4/85 (4.71%)
T 3	17/41 (41.46%)	1/41 (2.44%)	10/41 (24.39%)	13/41 (31.71%)

Considering **Time**, Table 6 shows the average and median time needed to create the model for every treatment. No statistically significant differences were found between the different treatments. Vocabulary support therefore does not incur time overhead during model creation, although the participants were not trained in using a vocabulary and had to deal with the overhead of searching through the ESO and selecting concepts as labels for modelling elements (rather than freely writing a label).

Table 6: Time needed for model creation

	T 1	**T 2**	**T 3**
Average time needed	11.52 min	10.70 min	11.20 min
Median time needed	11.20 min	10.25 min	9.60 min

The results for **Perceived ease of use** (PEOU), **Perceived usefulness** (PU) and **Intent of user** (IU) are summarized in Table 7, presenting averages of the post-survey Likert scale scores (1-5), in which a lower score is better for PEOU, and a higher score is better for PU and IU. The results show that for PEOU, Treatment 3 scores statistically significantly better – albeit only slightly – than Treatment 1. Regarding PU, Treatment 3 scores slightly better (statistically significant) than Treatment 1, and Treatment 2 scores slightly better than Treatment 1. For PEOU and PU, according to average and mode values, the differences are very small. Vocabulary support in itself was considered useful, as demonstrated by the higher PU score for Treatment 2 compared to Treatment 3. As hinted by informal user feedback, we see two explanations for the slightly worse user perception of Treatment 1. First, the previously mentioned technical problems were cited as the main cause of annoyance. Given the minimal differences, avoiding these would probably bring scores to a similar level as

Treatment 3. Second, in Treatment 1, participants indicated that the re-arranging of the list of suggestions for every modelling element according to relevance was annoying. Future work should test solutions that maintain the order of the suggestion list in Treatment 1, but indicate relevance in an alternative way (e.g. using colour coding). Given that the differences in PEOU and PU were minor, and taking into account the solvable technical difficulties with Treatment 1, we carefully conclude that there is no considerable additional frustration or errors accompanying the added vocabulary support to the modelling task. Finally, results for **Intention of use** (IU) (see Table 7) do not imply any statistical significant difference.

Table 7: Post-survey results for Ease of use (PEOU), Usefulness (PU), and Community acceptance (IU); cell values denote a Likert scale value (1-5), with 1 being best and 5 worst for PEOU, and 5 best and 1 worst and for PU and IU

	T 1		**T 2**		**T 3**		**Statistical analysis**
	mode	avg	mode	avg	mode	avg	
PEOU	2	3.09	2	3.2	2	2.93	T1⇔T3: significant
PU	4	3.09	4	3.67	4	3.23	T1⇔T3: significant T2⇔T3: significant
IU	4	2.98	3	2.90	4	3.11	

To summarize, supplying a modeller with ESO support has two main benefits: (1) it increases model interoperability by linking elements of the models with appropriate ESO concepts via annotations, and (2) it greatly enhances the consistency of labelling modelling elements, as the same label – and annotation with underlying ESO concept – is used for elements with intrinsically identical meaning. Furthermore, this experiment has demonstrated that the additional information and burden to find and select suitable ESO concepts during modelling does not require extra time, and does not impact on the modeller's acceptance of the modelling setup, nor does it have a negative influence on the validity of the models. However, the models created with vocabulary support were not as complete as those created by means of Treatment 3. This can be attributed to the fact that participants concentrated on finding the appropriate vocabulary rather than on creating complete models. The user

perception of our method was slightly worse compared to regular modelling. Feedback in the post-survey indicates that this was probably caused by technical problems with the tool. For user perception, keeping a stable order in the suggestion list may have a positive influence for CMOE+BPMN. These issues will be tackled in follow-up studies.

Finally, although vocabulary support has shown to be useful, the differences between CMOE+BPMN and (only) vocabulary support are mixed. Some positive effects of the alphabetically ordered vocabulary (Treatment 2) may be neutralized or reversed when a larger, more complex model and a more extensive ESO are used, as a greater variety of ESO concepts needs to be found in a larger amount of ESO concepts. The above-mentioned improvements to our method are expected to further tilt the scale in favour of CMOE+BPMN.

6 Conclusions and Future Work

This article introduces the recommendation-based conceptual modelling and ontology evolution Framework (CMOE+), with two main objectives: (1) to solve the interoperability problem across models by facilitating the creation of different types of conceptual models based on concepts from the ESO, and (2) to stimulate ESO evolution based on conceptual modelling feedback. The ESO documents and disambiguates the terms used within the enterprise and the relations between those terms, and is thus perfectly suited as a semantic basis for model creation in order to improve model interoperability and enable automatic integration and querying across models. On the other hand, the framework exploits valuable information generated during model creation to maintain and allow the ESO to evolve, as to keep it in sync with newly emerging and evolving needs of the enterprise. As such, the framework establishes a symbiosis between conceptual modelling and ontology evolution within an enterprise.

The framework is instantiated for the BPMN modelling language in a recommendation-based process modelling method (CMOE+BPMN). This instantiation

focuses on the modelling aspect of our framework, and shows how the ESO can be used during BPMN model creation to generate recommendations and annotate BPMN models. CMOE+BPMN supports setting up the ESO, analysing the selected modelling language, developing recommendation-based services, and extracting feedback. It was implemented as a plug-in that extends the Eclipse BPMN2 modeller, and was validated in an extensive exploratory experiment including 140 business students. The experiment showed some promising results: the use of an ESO vocabulary during modelling indeed results in more consistent labelling of modelling elements and does not incur any time overhead. What is more, users have the intention to use the method. Improvements can be made regarding user perception, which currently shows mixed signals, and model completeness, which could be improved as far as complete models are concerned.

Future research will aim at improving CMOE+BPMN and associated modelling tool to obtain better perceived usefulness and model completeness. If technical problems with the tool are overcome, order-invariant label suggestions are provided and more complex models and ESO are used, we expect the recommendation-based modelling method to be more advantageous than vocabulary-assisted modelling. On a broader scale, we have now finalized the instantiation of our method for requirements engineering using i* (Yu 1997), thus proving its wider applicability. Experiments to validate the i* instantiation are underway. Finally, we aim to exploit the modelling feedback, which has already been gathered and (manually) verified to be useful, in a more formal framework, through a community-based ontology evolution approach.

ACKNOWLEDGEMENT

The ontology-driven conceptual modelling research under the supervision of Frederik Gailly is funded by the National Bank of Belgium. Since November 2015, Sven Casteleyn is funded

by the Ramón y Cajal Programme of the Spanish government, grant number RYC-2014-16606.

REFERENCES

Abramowicz W, Filipowska A, Kaczmarek M, Kaczmarek T (2007) Semantically enhanced Business Process Modelling Notation. CEUR Workshop Proc. 251:

Almeida JP, Guizzardi G (2013) An ontological analysis of the notion of community in the RM-ODP enterprise language. Comput Stand Interfaces 35:257–268. doi: 10.1016/j.csi.2012.01.007

Becker J, Breuker D, Pfeiffer D, Räckers M (2009a) Constructing comparable business process models with domain specific languages – an empirical evaluation. Proc 17th Eur Conf Inf Syst 1–13.

Becker J, Delfmann P, Herwig S, Lis L, Stein a (2009b) Towards increased comparability of conceptual models-enforcing naming conventions through domain thesauri and linguistic grammars. ECIS 2009 Proc 1–13.

Becker J, Pfeiffer D, Falk T, Räckers M (2010) Semantic Business Process Management Handbook on Business Process Management 1. In: vom Brocke J, Rosemann M (eds)Springer Berlin Heidelberg, pp 187–211

Blomqvist E (2005) Fully automatic construction of enterprise ontologies using design patterns: Initial method and first experiences. Lect Notes Comput Sci 3761:1314–1329.

Born M, Dörr F, Weber I (2007) User-friendly semantic annotation in business process modeling. Web Inf. Syst. Eng. 2007 Work. Springer, pp 260–271

Cabral L, Norton B, Domingue J (2009) The Business Process Modelling Ontology. Proc. 4th Int. Work. Semant. Bus. Process Manag. ACM, New York, NY, USA, pp 9–16

Davis FD (1989) Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. Mis Q 13:319–340.

Delfmann P (2009) Unified Enterprise Knowledge Representation with Conceptual Models - Capturing Corporate Language in Naming Conventions. ICIS 2009 Proc 1–16.

Di Francescomarino C, Tonella P (2009) Supporting ontology-based semantic annotation of business processes with automated suggestions. In: Halpin T, Krogstie J, Nurcan S, Proper E, Schmidt R, Soffer P, Ukor R (eds) Lect. Notes Bus. Inf. Process. Springer Berlin Heidelberg, pp 211–223

Evermann J, Wand Y (2005) Toward formalizing domain modeling semantics in language syntax. Ieee Trans Softw Eng 31:21–37.

Fill H-G (2012) An Approach for Analyzing the Effects of Risks on Business Processes Using Semantic Annotations. ECIS 2012 Proc.

Fill H-G (2011a) Using Semantically Annotated Models for Supporting Business Process Benchmarking. In: Grabis J, Kirikova M (eds) Perspect. Bus. Informatics Res. Springer Berlin Heidelberg, pp 29–43

Fill H-G (2011b) On the Conceptualization of a Modeling Language for Semantic Model Annotations. In: Salinesi C, Pastor O (eds) Adv. Inf. Syst. Eng. Work. Springer Berlin

Heidelberg, pp 134–148

Francescomarino C Di, Ghidini C, Rospocher M, Serafini L, Tonella P (2011) A framework for the collaborative specification of semantically annotated business processes. J Softw Maint Evol Res Pract 23:261–295. doi: 10.1002/smr

Gailly F (2016) Recommendation-based Conceptual Modeling and Ontology Evolution (CMOE+) Java Tool. doi: 10.5281/zenodo.167132

Geerts GL, McCarthy WE (1999) An Accounting Object Infrastructure for Knowledge Based Enterprise Models. IEEE Intell Syst Their Appl 14:89–94.

Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z (2014) HermiT: An OWL 2 Reasoner. J Autom Reason 53:245–269. doi: 10.1007/s10817-014-9305-1

Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U (2008) OWL 2: The next step for OWL. Web Semant 6:309–322. doi: 10.1016/j.websem.2008.05.001

Gregor S, Hevner AR (2013) Positioning And Presenting Design Science Research For Maximum Impact. MIS Q 37:337-A6.

Guarino N (1998) Formal Ontology and Information Systems. Int. Conf. Form. Ontol. Inf. Syst. IOS Press, Trento, Italy, pp 3–15

Guarino N, Welty C (2002) Evaluating ontological decisions with ontoclean. Commun Acm 45:61–65.

Guizzardi G (2013) Ontology-Based Evaluation and Design of Visual Conceptual Modeling Languages. Domain Eng Prod Lines, Lang Concept Model 345. doi: 10.1007/978-3-642-36654-3

Guizzardi G, Wagner G (2011) Can BPMN Be Used for Making Simulation Models? In: Barjis J, Eldabi T, Gupta A (eds) Enterp. Organ. Model. Simul. Springer Berlin Heidelberg, pp 100–115

Guizzardi G, Wagner G, Almeida JPA, Guizzardi RSS (2015) Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. Appl Ontol 10:259–271. doi: 10.3233/AO-150157

Hahn A (2005) Integration verteilter Produktmodelle durch Semantic-Web-Technologien. Wirtschaftsinformatik 47:278–284. doi: 10.1007/BF03254915

Harzallah M, Berio G, Opdahl AL (2012) New perspectives in ontological analysis: Guidelines and rules for incorporating modelling languages into UEML. Inf Syst 37:484–507. doi: 10.1016/j.is.2011.11.001

Hepp M, Leymann F, Domingue J, Wahler A, Fensel D (2005) Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. IEEE ICEBE. IEEE, Beijing, CHINA, pp 535–540

Hepp M, Roman D (2007) An Ontology Framework for Semantic Business Process Management. Wirtschaftsinformatik 2007

Hevner AR, March ST, Park J, Ram S (2010) Design Science in Information Systems Research. MIS Q 28:75–105.

Hofferer P (2007) Achieving Business Process Model Interoperability Using Metamodels and Ontologies. Eur. Conf. Inf. Syst. pp 1620–1631

Leutgeb A, Utz W, Woitsch R, Fill H-G (2007) Adaptive Processes in E-Government - A

Field Report about Semantic-Based Approaches from the EU-Project FIT. ICEIS 2007 - Proc Ninth Int Conf Enterp Inf Syst Vol EIS, Funchal, Madeira, Port June 12-16, 2007 264–269.

Lindland OI, Sindre G, Solvberg A (1994) Understanding Quality in Conceptual Modeling. IEEE Softw 11:42–49.

Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38:39–41. doi: 10.1145/219717.219748

Moody D (2003) The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods. 11th Eur. Conf. Inf. Syst. ECIS 2003

Mylopoulos J (1992) Conceptual Modelling and Telos. In: Loucopoulos P, Zicari R (eds) Concept. Model. databases, CASE. Wiley, pp 49–68

OMG (2006) Business Process Modeling Notation Specification (dtc/06-01-01).

Opdahl AL, Berio G, Harzallah M, Matulevicius R (2012) An ontology for enterprise and information systems modelling. Appl Ontol 7:49–92. doi: 10.3233/ao-2011-0101

Opdahl AL, Henderson-Sellers B (2002) Ontological Evaluation of the UML Using the Bunge–Wand–Weber Model. Softw Syst Model 1:43–67.

Pfeiffer D (2007) Constructing comparable conceptual models with domain specific languages. Proc 15th Eur Conf Inf Syst ECIS 2007 876–888.

Pinggera J, Zugal S, Weber B (2010) Investigating the process of process modeling with cheetah experimental platform- Tool paper. CEUR Workshop Proc. pp 13–18

Pittke F, Leopold H, Mendling J (2013) Spotting terminology deficiencies in process model repositories. Lect. Notes Bus. Inf. Process. pp 292–307

Ruy FB, Reginato CC, Santos VA, Falbo RA, Guizzardi G (2015) Ontology Engineering by Combining Ontology Patterns. pp 173–186

Suárez-Figueroa MC, Gómez-Pérez A, Motta E, Gangemi A (2012) Ontology Engineering in an Networked World. Springer Berlin / Heidelberg

Thomas O, Fellmann M.A. M, Fellmann M.A M (2009) Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. Bus Inf Syst Eng 1:438–451. doi: 10.1007/s12599-009-0078-8

Uschold M, King M, Moralee S, Zorgios Y (1998) The Enterprise Ontology. Knowl Eng Rev Spec Issue Putt Ontol to Use 13:31–89.

Winkler W (1990) String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage.

Yu ESK (1997) Towards modelling and reasoning support for early-phase requirements engineering. Requir. Eng. 1997., Proc. Third IEEE Int. Symp. pp 226–235

Appendix A: Unified Foundational Ontology

To exemplify the recommendation-based business process modelling, a subset of UFO (Figure 1) was used. The top-level element is a *Universal*. It represents a classifier that classifies a set of real world individuals and can be of four kinds: *Event Type, Object type, Quality Universa*l and *Relator Universal*:

- An object type is existentially independent universal which can be further specialized in a *Mixin type* and a *Sortal type*. A *Sortal type* supplies a principle of identity to its instances, while instances of *Mixin type* do not carry identifiers, as for example, Colored object. *Sortal type* can be *Rigid* (base type) or *Anti-rigid* (role and phase types). Rigid sortal implies that every instance of this type is necessarily its instance in all occasions; if Lana is an instance of Person, she will always be an instance of Person, hence Person is a *rigid* sortal. At one point, Lana is an instance of Teenager, and as she grows, she will not fit under Teenager anymore and this will not change her identity. So, Teenager is an *anti-rigid* sortal. Teenager constitutes a stage of individual's life cycle, hence it belongs to *Phase type*. The last subtype of sortal is Role type. *Role type* stands for a role played by an individual, for instance secretary, doctor, etc.
- A *Quality universal* is instantiated by qualities possessed by Object types, such as color and temperature.
- A *Relator universal* classifies mediators that mediate two individuals, as for example, medical treatment mediates a hospital and a person. A such a Relator universal is an objectification of a Material relationship between two or more Universals.
- Finally, an *Event type* is instantiated by an event. Events, in contrast to objects, qualities and relators are individuals composed of temporal parts, they happen in time, in the sense that they extend in time and accumulate temporal parts.

For a full explanation of UFO we refer to (Guizzardi et al. 2015).

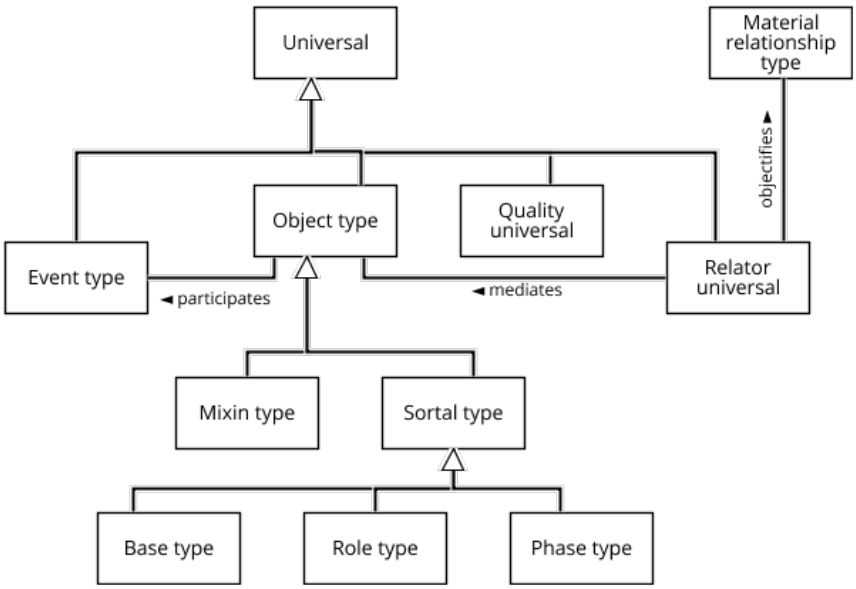

Figure 1: Fragment UFO

Appendix B: Correspondence between ESO and UFO

ESO concept	UFO	ESO concept	UFO
AddedValue	Quality_Universal	Loan	RelatorUniversal
Administrative	Role_Type	LoanApplication	RelatorUniversal
Asset	Mixin_Type	LoanApplication Accepted	EventType
Branch	Base_Type	LoanApplication Received	EventType
BuyCostProperty	Quality_Universal	LoanApplication Rejected	EventType
Capital	Base_Type	LoanApplication Verified	EventType
Channel	RelatorUniversal	LoanCommitment	RelatorUniversal
Collection	QualityUniversal	Login	QualityUniversal
Commercial	RoleType	MortgageLoan	RelatorUniversal
Company	BaseType	MortgageTaxation	QualityUniversal
Corporative	BaseType	Name	QualityUniversal
CreditHistory	RelatorUniversal	Payment	Relator Universal
Currency	BaseType	Person	BaseType
CurrentMortgage Loan	RelatorUniversal	Product	MixinType
Customer	MixinType	ProductRate Application	RelatorUniversal
DelayInterestRate	QualityUniversal	ProductRate ApplicationFixed	MixinType
Department	BaseType	ProductRate ApplicationMixed	MixinType
Document	BaseType	ProductRate Application Variable	MixinType
Employee	RoleType	ProofOfIncome	BaseType
EndingDate	QualityUniversal	PropertyAppraisal Report	BaseType
ExpirationDate	QualityUniversal	Quota	QualityUniversal
FutureMortgage Loan	RelatorUniversal	QuotaAfterRevision	QualityUniversal
HandlingCapital	QualityUniversal	RepaymentAbility	RelatorUniversal
HomeInsurance	QualityUniversal	RevisionTermNextService	QualityUniversal MixinType
Individuals	BaseType	SavingsAccount	BaseType
InitialPeriod	QualityUniversal	Service	MixinType
InitialQuota	QualityUniversal	ServiceContract ByCustomer in Chanel	MixinType
InterestDelay	QualityUniversal	SignalDateContract	QualityUniversal
InterestRateValue	QualityUniversal	SME	BaseType
InvestmentAccount	RelatorUniversal	SOHO	BaseType
InvestmentFund	RelatorUniversal	Staff	RoleType
Invoice	RelatorUniversal	StartingDate	QualityUniversal

Liability	RelatorUniversal	Term	QualityUniversal
LifeInsurance	RelatorUniversal	User	MixnType
		vBanking	BaseType

Appendix C: Case description

A person deciding to get a mortgage loan sends a loan application to the chosen branch of his/her bank. When the administrative employee working at that branch receives the loan application from the bank's customer, he starts making the decision on whether to grant the loan or not. The employee assesses the client's ability to repay the mortgage. If this analysis shows the applicant is not likely to repay the mortgage loan, his/her request is rejected. If the customer is found to be capable of repaying, the bank representative evaluates his/her assets (such as house and other properties). The employee then verifies whether the bank customer requested a home insurance or not. If the insurance was not requested, a loan acceptance notification is sent to the applicant. If the insurance is requested, the notification is sent together with a home insurance quota.

Appendix D: BPMN tool

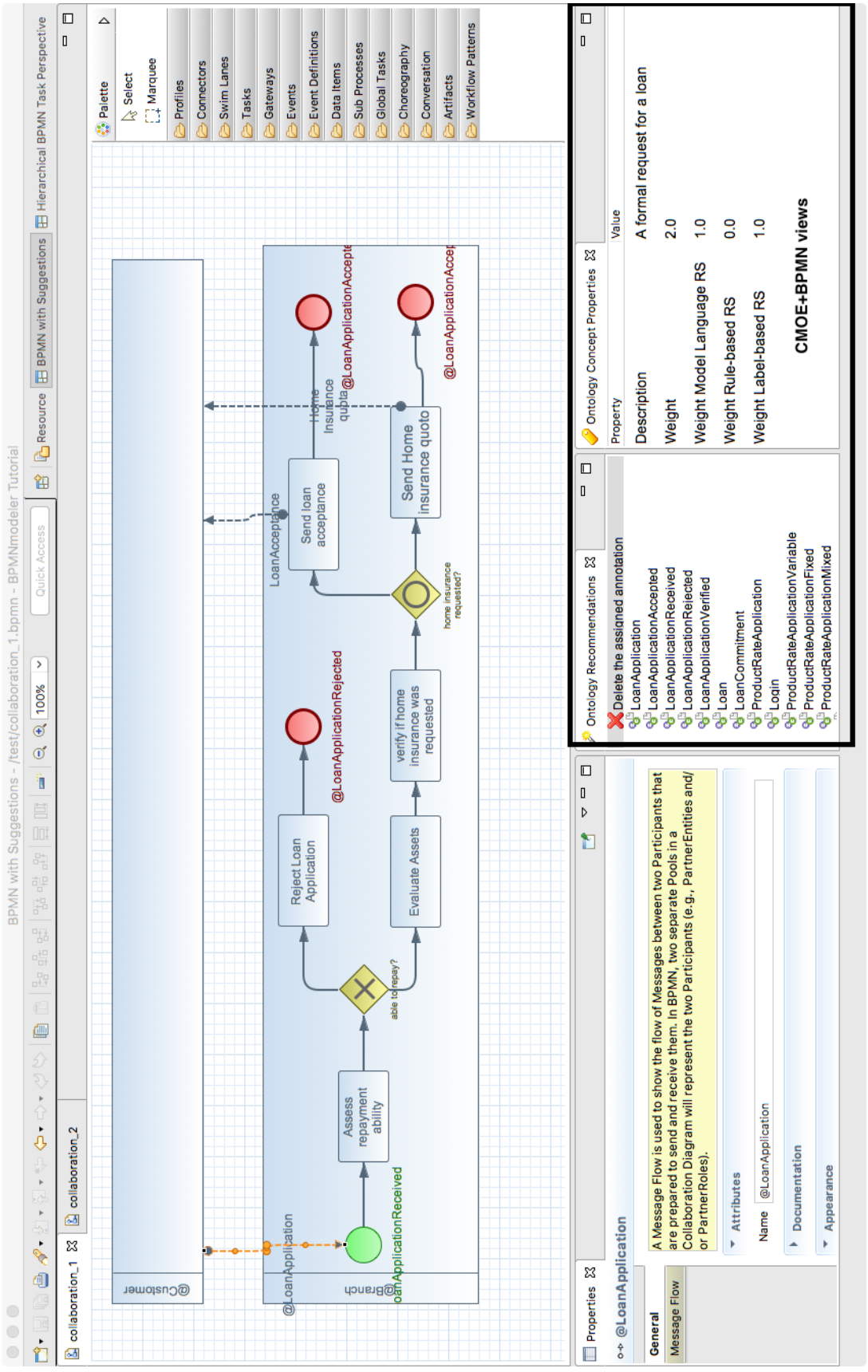

Figure 2: BPMN tool indicating the differences for different treatments in the experiments

Appendix E: Pre-survey

Q1: What is your gender?
Q2: Which study program are you following?
Q3: Did you have any BPMN training prior to attending the BPMN course? (yes/no)
Q4: Overall, I am familiar with Business Process Model and Notation (BPMN).

Q5: I feel competent in using BPMN for business process model creation.
The answers for the last two questions are on a likert scale from 1 (not familiar / competent) to 5 (very familiar / competent).

Appendix F: Post-survey

Questions of the post-survey classified according to the dependent variables to be measured:
PEOU1: I often made errors while modelling BPMN diagrams

PEOU2: I found it frustrating to model BPMN diagrams

PEOU3: I found it require a lot of mental effort to model BPMN diagrams

PU1: I was able to create BPMN models quickly

PU2: I was able to label BPMN elements easily

PU3: It was hard for me to find relevant domain concepts to use as a label for BPMN

elements

IU1: Overall, I found the given setup useful for BPMN model creation

IU2: I would definitely use the given setup for model creation

Appendix G: Reference Model

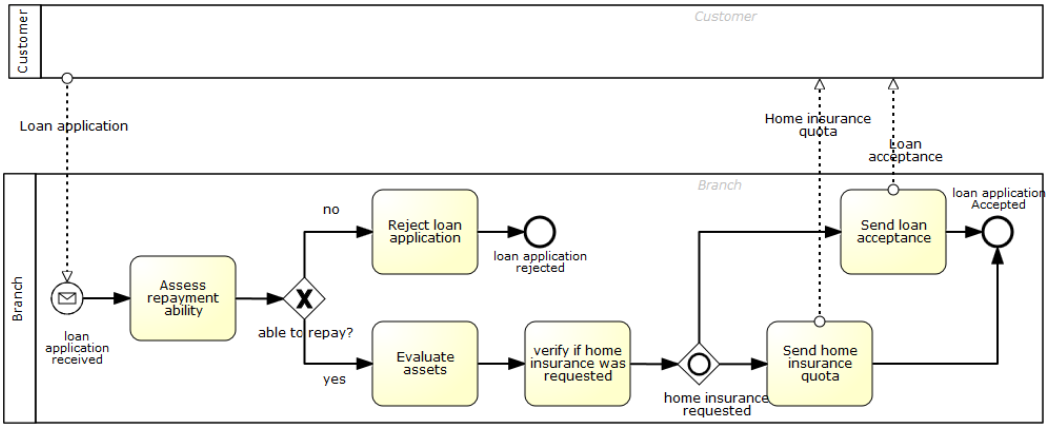

Figure 3: reference model