



Faculty of Economic and
Social Sciences and
Solvay Business School



Faculty of Economics
and Business
Administration

On the Symbiosis between Conceptual Modeling and Ontology Engineering: Recommendation-Based Conceptual Modeling

Dissertation submitted in fulfillment of the requirements of the degree
of Doctor of Business Economics

Nadejda Alkhaldi

PhD Candidate

*PhD program for joint supervision and awarding of a doctorate
diploma between the Vrije Universiteit Brussel and Ghent University*

Supervisor Vrije Universiteit Brussel: Prof. dr. Wouter Verbeke

Supervisor Ghent University: Prof. dr. Frederik Gailly

Supervisor Universitat Jaume I de Castelló: Prof. dr. Sven Casteleyn

Academic year: 2017 – 2018

Copyright © 2017 by Nadejda Alkhaldi

All rights are reserved. No part of this publication may be reproduced or transmitted in any form or by any means electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the author.

Examination Board

Prof. dr. Wouter Verbeke (VUB)
Prof. dr. Frederik Gailly (UGent)
Prof. dr. Sven Casteleyn (Universitat Jaume I de Castelló)
Prof. dr. Peter De Bruyn (VUB)
Prof. dr. Tias Guns (VUB)
Prof. dr. Geert Poels (UGent)
Prof. dr. Sergio de Cezare (University of Westminster)
Prof. dr. Monique Snoeck (KU Leuven)

Acknowledgement

I would like to show appreciation for my supervisors: Frederik Gailly, Sven Casteleyn, and Wouter Verbeke for their advice and support during the writing of this dissertation. I want to say thank you to my colleagues at the VUB and UGent for fun conversations over lunch, and emotional support during the time of hardship. Additionally, I want to thank my friends who helped me to get the most of my stay in Belgium, and were there for me when needed. Finally, thank you to my parents and siblings for their support and for having trust in me even when I did not seem to find it within myself.

Table of Contents

Examination Board	2
Acknowledgement	3
List of Figures	7
List of Tables	8
Abstract	10
Introduction	14
1.1 Background	14
1.1.1 Ontology	14
1.1.2 Ontology Engineering	16
1.1.3 Conceptual Modeling	19
1.1.3.1 Business process modeling	20
1.1.3.2 Goal modeling	23
1.2 Conceptual Model vs Ontologies	24
1.3 Problem Description	27
1.4 Research Goals	30
1.5 Methodology	31
1.6 Publications	33
1.7 Thesis Structure	34
Recommendation-Based Conceptual Modeling and Ontology Evolution Framework	36
2.1 Semantic Alignment of Conceptual Models	36
2.2 Requirements for CMOE+	40
2.3 Development of CMOE+	43
2.3.1 Ontology Evolution Cycle:	45
2.3.1.1 Ontology setup	46
2.3.1.2 Ontological storage and recommendation services	47
Recommendation services	48
2.3.1.3 Community – based ontology feedback evaluation	55
2.3.1.4 Ontology evolution	56
2.3.2 Conceptual Modeling Cycle	57
2.3.2.1 Ontological analyses of the modeling language	57
2.3.2.2 Conceptual model creation	60
2.3.2.3 Conceptual model evaluation	63
2.4 Correspondence Between CMOE+ Requirements and Phases	65
2.5 Conclusion	67

Recommendation-Based Process Modeling.....	72
3.1 Introduction	72
3.2 Recommendation-Based Conceptual Modeling and Ontology Evolution (CMOE+) Framework.....	74
3.2.1 Ontology Setup.....	74
3.2.2 Ontological Analysis of the Conceptual Modeling Language	75
3.2.3 Ontology Storage and Recommendation Services.....	76
Recommendation Services.....	78
3.2.4 The Conceptual Model Creation Phase.....	81
3.3 Recommendation-Based Business Process Modeling (CMOE+BPMN)	82
3.3.1 Ontology Storage	84
3.3.2 Recommendation Services.....	85
3.4 Evaluation of CMOE+BPMN	86
3.4.1 Experimental Design	87
3.4.2 Experiment Measures	88
3.4.3 Experimental Results.....	89
3.5 Conclusions and Future Work	93
Aligning I* Models and BPMN Models Using the CMOE+ Framework	96
4.1 Aligning Goal and Process Models	96
4.2 CMOE+ I* Tool.....	99
4.3 Extending CMOE+BPMN with I*-specific Recommendation Rules.....	102
4.3.1 Model Repository	102
4.3.2 Rule-Based Recommendation Services.....	103
4.4 Demonstration	108
4.5 CMOE+X Tool Architecture	111
4.6 Conclusion	112
Conclusions and Future Work	116
5.1 Research Results	116
5.2 Contributions	121
5.3 Limitations and Future Work	122
Acronyms	126
References	128
BPMN Constructs	138
I* Constructs.....	141
Unified Foundational Ontology (UFO)	144
Correspondence between ESO and UFO	146
Loan Application Case Description	148

Pre-survey	149
Post-survey.....	150
Reference Model.....	151

List of Figures

Figure 1: Main activity categories within ontology engineering process. Adapted from (Gomez - Perez et al. 2003).....	17
Figure 2: Differences between ontology and conceptual model taken from (Fonseca & Martin 2007)	27
Figure 3: Regulative cycles of the PhD inspired by (Wieringa & Heerkens 2006).....	32
Figure 4: Different phases of CMOE+, and the interaction between them	44
Figure 5: IS research framework followed during the development of CMOE+.....	45
Figure 6: Recommendation services	50
Figure 7: Model language-based recommendation service.....	52
Figure 8: Label-based recommendation service	53
Figure 9: Rule-based recommendation service	55
Figure 10: Recommendation-based Conceptual Modeling and Ontology Evolution (CMOE+) Framework	75
Figure 11: Ontologies within CMOE+ framework	78
Figure 12: BPMN tool	83
Figure 13: BPMN meta-model.....	85
Figure 14: Ontology Recommendation view (Left) and Ontology Concept Properties view (Right)	86
Figure 15: I* goal dependency	100
Figure 16: Screenshot of CMOE+i*	102
Figure 17: An excerpt of an i* model created by the CMOE+i* tool (on the left), and the corresponding OWL representation stored in the model repository (on the right)	103
Figure 18: Graphical representation of Rule 3	106
Figure 19: Graphical representation of Rule 4	107
Figure 20: Graphical representation of Rule 5	108
Figure 21: Simplified process model for the emergency department.....	110
Figure 22: Components of CMOE+ tool. The components highlighted in red need to be altered for every modeling language	112
Figure 23: The final version of CMOE+.....	114
Figure 24: Phases of CMOE+ which are elaborated in detail within this dissertation	119

List of Tables

Table 1: Existing approaches to enhancing semantic alignment of models using ontology	38
Table 2: Requirements for the proposed framework	65
Table 3: Summary of different phases of CMOE+	67
Table 4: Correspondence between BPMN and UFO	85
Table 5: SWRL rules used by the rule-based recommendation service.....	87
Table 6: Results of semantic quality evaluation model annotation.....	90
Table 7: Naming for BPMN elements with underlying meaning "customer". Columns are modeler-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning "customer"	91
Table 8: Naming for BPMN elements with underlying meaning "loan application". Columns are model-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning "loan application"	91
Table 9: Time needed for model creation.....	91
Table 10: 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 for PU and IU	92
Table 11: Ontological analyses of i*	99

Abstract

Within an enterprise, different conceptual models, such as process, data, and goal models, are created by various stakeholders. These models are fundamentally based on similar underlying enterprise (domain) concepts, but they have a different focus, are represented using different modeling languages, take different viewpoints, utilize different terminology, and are used to develop different enterprise artefacts (such as documents, software, databases, etc.); therefore, they typically lack consistency and alignment. Another issue is that modelers have different vocabulary selections and different modeling styles. As a result, the enterprise can find itself accumulating a pile of models which cover similar aspects in different manners. Those models are not machine-readable and cannot be processed automatically. Enterprise-Specific Ontologies (ESOs) aim to solve this problem by serving as a reference during the conceptual model creation. Using such a shared semantic repository makes conceptual models semantically aligned and facilitates model integration. However, managing those ontologies is complicated; an enterprise is an evolving entity, and as it changes, the ESO might become outdated.

This research aims to assist modelers within an enterprise to create aligned conceptual models by basing them on the ESO. To do so, the ESO has to be encapsulated and presented to the modelers through an interface which does not require advanced knowledge on ontologies. Since the ESO can be very extensive in size, the proposed solution must incorporate recommendation services to support the modeler in viewing the ESO concepts in an ordered and supportive manner.

During the years of research dedicated to this dissertation, the Recommendation-Based Conceptual Modeling and Ontology Evolution (CMOE+) framework was developed. This framework establishes a symbiotic relationship between the Ontology engineering and the Conceptual modeling fields. CMOE+ consists of two cycles: the Ontology Evolution cycle and the Conceptual Modeling cycle. The *Ontology Evolution cycle* is responsible for setting up the initial version of the ESO and updating it as the knowledge within the enterprise is

updated. Additionally, this cycle encapsulates recommendation services to perform ontology look-up and to present the most relevant ESO concepts in support of the modeler. The *Conceptual Modeling cycle* is responsible for the creation of conceptual models in different modeling languages based on the ESO. CMOE+ was developed based on requirements identified as a result of a literature review and a case study. The development process follows the Design Science Research Methodology (DSRM). After the initial version of CMOE+ was put forward, the focus was narrowed towards the recommendation-based conceptual modeling part of CMOE+, and continued to gradually improve the framework in iterations until it reached its current state. The Ontology Evolution Cycle is not fully addressed within this dissertation. CMOE+ is a general framework which is not bound to a particular modeling language. Moreover, different phases of CMOE+ can be adapted to accommodate the particular needs of the enterprise.

In order to demonstrate the performance and usability of CMOE+, it was exemplified for process modeling using BPMN and goal modeling using i*. This thesis presents a detailed instantiation, and explains which steps are to be performed in order to instantiate CMOE+ for other modeling languages. In order to evaluate the process modeling instance of CMOE+, a CMOE+BPMN tool was implemented. This tool incorporates a BPMN modeler, facilitates storage and access of the ESO, and includes all algorithms functioning within CMOE+ for the BPMN modeling language (as some of the algorithms are language dependent). Next, CMOE+ was exemplified using the i* goal modeling language. Finally, we tested the ability of CMOE+ to perform alignment between i* and BPMN models, in order to show that CMOE+ is indeed beneficial in achieving interoperability among models created in different modeling languages and covering distinct aspects of the enterprise.

The main contributions of this thesis are summarized as following:

1. Proposing a comprehensive framework for conceptual model creation based on an ESO, while simultaneously updating the ESO based on the knowledge captured from those models. This framework is language-independent, and it improves semantic alignment of models already during model creation. Additionally, this framework applies the theory of ontological analyses of the modeling language in practice.
2. Establishing recommendation services which access the ESO, scan it, and, by using a reconfigurable set of algorithms determine the ESO concepts with the most potential relevance for the modeler during model creation.
3. Implementing a prototype of the CMOE+BPMN tool which incorporates all the features addressed in this thesis. This tool was evaluated with students using Moody's Method Evaluation Model (MEM). Moreover, this work explains what needs to be adjusted while instantiating CMOE+ for other modeling languages.

4. Establishing semantic alignment between goal models in i* and process models in BPMN. This work demonstrates how CMOE+ framework is used to create BPMN models which are semantically aligned to i* models previously created within an enterprise.
5. Facilitating semantic annotation of different types of conceptual models. CMOE+ framework allows annotating modeling element with concepts from the enterprise-specific ontology. The annotation is performed during model creation.

1

Introduction

This chapter provides the background on which the research is built. It clarifies the main concepts and approaches featuring in the thesis including Ontology, Ontology Engineering, and Conceptual Modeling. It explains the difference between ontologies and conceptual models within the context of this thesis. Further, the existing problem, research goals are described. An overview of the followed research methodology is provided. Finally, this chapter lists the papers published within the course of this PhD, and concludes with giving an overview of the structure.

1.1 Background

This PhD thesis touches upon two main topics: Ontology Engineering and Conceptual Modeling. It aims at exploring the relationship between both concepts and creating an environment where both of them can coexist and greatly benefit from each other. This first section defines both concepts, and clarifies the difference between the two.

1.1.1 Ontology

Ontologies originate in the philosophy discipline, but nowadays they are widely used in artificial intelligence, computer science, knowledge engineering, and many other fields. Given this popularity, many definitions of ontology were suggested by researchers. One of the first definitions of Ontology was put forward by Neches and his colleagues (Neches et al. 1991, page 4) and states: “An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary”. The most cited definition of the ontology was provided a few

years later by Gruber: "Ontology is an explicit representation of conceptualization" (Gruber 1993, page 1). Ontologies are sometimes confused with taxonomies (Studer et al. 1998). However, an ontology captures more domain semantics than pure taxonomical arrangement of domain concepts. In addition to taxonomical relationships, the ontology adds relationships, properties of concepts, axioms and constraints on concepts' usage.

Through the years, ontologies were built and represented using a variety of representation techniques. In the beginning of 90s, AI modeling techniques based on frames and first order logic were used (Lenat and Guha, 1990). In the same field, Gruber (1993) suggests using five main components: *classes* which represent ontological concepts, *relations* standing for associations among the concepts, *functions* are special types of relations where one element depends on another, *formal axioms* allow modeling sentences which are always true. And finally, *instances* represent individuals in the ontology. Later, when the emergence of semantic web, description logics based techniques (Baader et al, 2003) gained popularity for ontology representation, and various description logic languages were developed such as Ontology Inference Layer OIL (Fensel et al. 2000) and Web Ontology Language OWL (Bechhofer et al. 2003). Description logics techniques permit formalizing ontologies using three components: *concepts* stand for ontological concepts, *roles* which are binary relations among ontological concepts, and *individuals* represent instances of ontological concepts. Representing an ontology by means of description logics allows the separation of terminological knowledge from assertional knowledge by using respectively the TBox and ABox mechanisms of description logics theory. TBox is responsible for presenting terminological knowledge of the ontology covering concepts with their definitions, formal properties and roles, while ABox is used to represent individuals of concepts in a particular domain.

Apart from AI related techniques, ontologies can be represented using software engineering techniques, such as the Unified Modeling Language (UML) (OMG 2007). The UML representation is easy to understand for people without computer science background, and there are many Computer Aided Software Engineering CASE tools available. UML allows modeling ontological concepts, their instances, and relationships between the concepts. Database technology can also be used for modeling ontologies, such as Entity / Relationship diagrams (ER) (Chen 1976).

Ontology researchers make a distinction between ontologies at three different levels: core ontologies, domain ontologies and application ontologies (Guarino 1998). A core (high-level) ontology describes general concepts that are independent of a specific domain. Examples are the Suggested Upper Merged Ontology (SUMO) (IEEE Standard Upper Ontology Working Group n.d.), Descriptive Ontology for Linguistic and Cognitive Engineering DOLCE (Gangemi et al. 2002) and the Unified Foundational Ontology (UFO) (Guizzardi & Wagner 2010c). A domain ontology describes the concepts, relations and axioms specific to a particular

domain (van Heijst et al. 1997) (like medicine, or automobiles) by specializing the terms introduced in the core ontology. Finally, an application ontology adds specificities to the domain ontology which are only relevant for the application considered, but are not shared across the entire domain.

An enterprise ontology is considered as a specific type of domain ontologies. It describes shared concepts and relationships across an enterprise. In literature, the term “enterprise ontology” refers to an ontology that has its origins in economic and business theory, and that is not specific to a particular business. Well-known examples are the Enterprise Ontology (Uschold et al. 1996), and Toronto Virtual Enterprise Ontology TOVE (Fox 1992).

The research presented here benefits from ontologies at various levels but the main concern and the final product of this thesis is an Enterprise-Specific Ontology (ESO). ESOs are a special type of 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).

1.1.2 Ontology Engineering

Gomez – Perez et al define *ontology engineering* as “the set of activities which concerns the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies” (Gomez - Perez et al. 2003). Building and maintaining ontologies is a tedious and complex process. Following a methodologic framework can considerably facilitate this process by adding structure, decomposing the process into manageable tasks and clarifying the responsibilities of each participant (Simperl & Tempich 2006). One of the first researchers proposing such a framework was Fernandez – Lopez and his colleagues in 1997 (Fernández-López et al. 1997). They offered a methodology for ontology construction – METHONTOLOGY - which was based on IEEE standards for software development (IEEE 1997). Later, the literature identified three activity categories crucial within the ontology development process, especially when the cooperating team is distributed over geographical locations. According to (Gomez - Perez et al. 2003) those categories are ontology management, ontology development, and ontology support as presented in Figure 1 below.

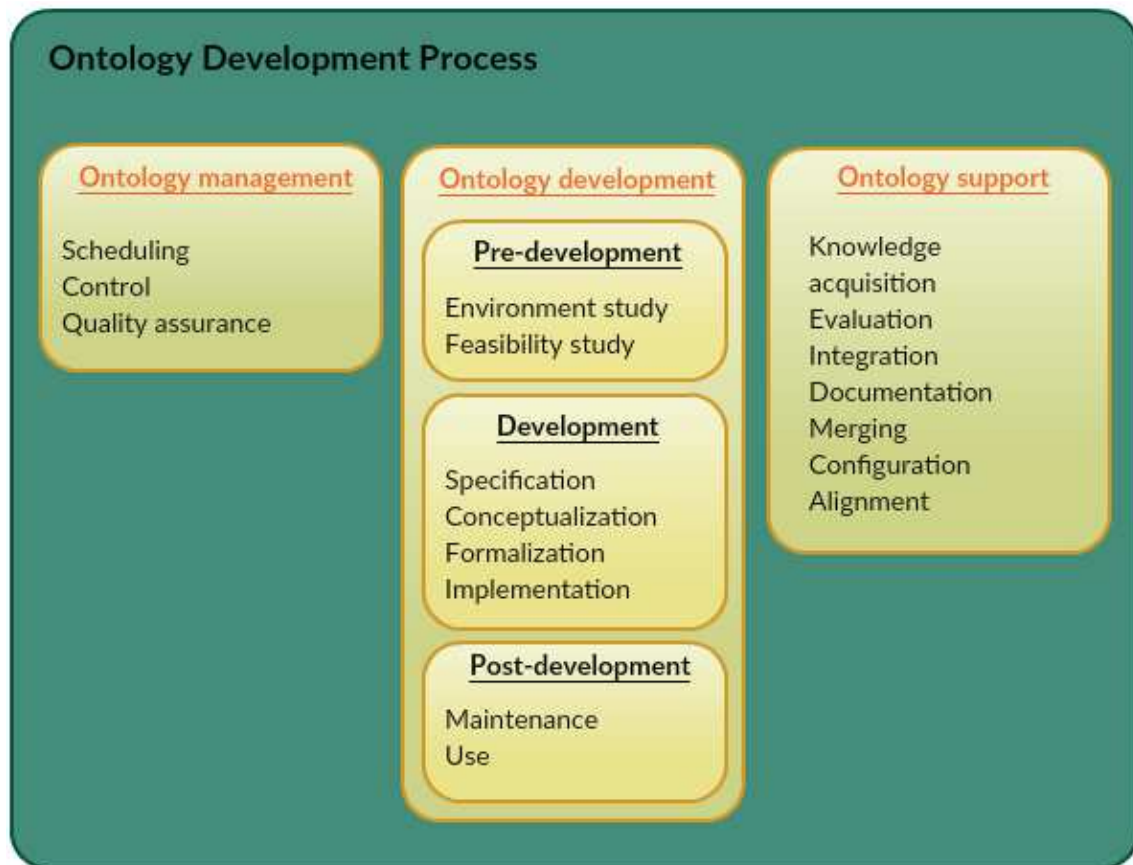


Figure 1: Main activity categories within ontology engineering process. Adapted from (Gomez - Perez et al. 2003)

The **Ontology management** category constitutes three activities: Scheduling, Control, and Quality assurance. The *Scheduling* activity prescribes the tasks to be performed, their order, and time and resources required to accomplish each task. The *Control* activity is responsible for monitoring the scheduled tasks and ensuring that everything runs according to the predefined schedule. The last activity, *Quality assurance*, ensures that the quality of the final product corresponds to quality standards.

The activities within the **Ontology development** category are grouped into Pre – development, Development, and Post – development. During *Pre – development* activities an environmental study is performed to capture information about the platform and the applications where the ontology will be integrated. The *Development* activities include documenting intended uses of the ontology and its end – users, structuring domain knowledge as meaningful models, formalizing those models, and implementing them into an ontology language. The *Post – development* activities include ontology maintenance, update and use.

The **Ontology support** category constitutes of activities related to Knowledge acquisition, Evaluation, Integration, Merging, Alignment, Documentation, and Configuration. Those activities are performed in parallel with ontology development activities.

Various methods and methodologies put forward in the literature are not only concerned with ontology development from scratch, but also with utilization and reuse of what is available. Ontology development is an effort intensive process, hence complete or partial reuse of existing products is absolutely necessary. An example of such a methodology is the one proposed by (Gómez-Pérez & Rojas-Amaya 1999). If the knowledge to be reused is presented in several ontologies, those ontologies can be either merged or aligned. Noy and Musen (Noy & Musen 1999) define **ontology alignment** as constructing mappings (or links) between two ontologies in a way that both ontologies are preserved. **Ontology merging** is defined as a process of generating a new ontology from the original ones. Ontology reuse started to capture the attention of scientists. Recently, Katsumi and Gruninger published their work (Katsumi & Gruninger 2016) where they define the boundaries of ontology reuse and its possibilities.

If an ontology engineer has decided to build an ontology from scratch, he can benefit from ontology learning techniques. **Ontology learning** methods, such as (Kietz et al. 2000), are used to minimize efforts necessary for acquiring the knowledge for building ontologies. Ontology learning techniques partially automate knowledge acquisition process by means of natural language analyses or machine learning techniques. Maedche and Staab (2000) distinguish four ontology learning approaches. (1) Learning from text, which utilizes text corpora (2) learning from instances where concrete instances are used (3) learning from schemata such as relational database, XML schemas and data models (4) learning from interoperability where semantic mappings between elements of two ontologies are learnt.

Following from the fact that an ontology should be shared by a community, the members of a community also play an important role in the development process. Collaborative ontology engineering includes methods and tools that are designed to support a decentralized group of stakeholders with varying level of skills, experience and responsibility, and divergent agendas, distributed over geographical locations to reach consensus in an incremental fashion. (Simperl & Luczak-Rösch 2014). To guide the collaboration process during the ontology development and give it direction, some methodologies utilize the notion of a specialized *process model* which distributes the role and prescribes policies and tools to be used. Such a process model is found in (Vrandecic et al. 2005) and (Holsapple & Joshi 2002). Collaborative ontology engineering is often an iterative process. Furthermore, several community members at different locations might access the ontology concurrently. This is managed through special versioning software such as (Noy & Musen 2004) and (Luczak-Rösch et al. 2010). In order to structure community negotiation and interaction, methodology such as the Delphi technique (Gupta & Clarke 1996) and Nominal group (Dunnette et al. 1963) are incorporated in the process. Those methodologies are mediated by communication channels such as elementary forums and chats (Tudorache et al. 2008). A group of researchers also attempted to use wikis in collaborative ontology engineering settings. IkeWiki, developed by Schaffert, is a wiki system which allows users to annotate

pages and links between them using semantic OWL and RDF thereby formalizing informal text into formal ontologies (Schaffert 2006). Another example is the OntoWiki approach proposed by Hepp et al (2006) which allows community members to create a URI for a concept, describe and refine its definition, and link it to concepts already available on Wikipedia.

The first comprehensive methodology for collaborative ontology engineering was proposed by Holsapple and Joshi (2002). According to this methodology, an initial ontology was constructed by integrating existing ontologies in the domain. Later the initial ontology was subject to community discussion and revision. Nowadays various collaborative ontology engineering methodologies are available for different purposes. Moor et al propose a methodology specialized in interorganizational settings with many pre-existing ontologies, and ill-defined changing requirements. Hereby, it supports construction of a sequence of increasingly complex versions of interrelated ontologies over time (Moor et al. 2006). Missikoff et al describe an iterative methodology to construct domain-specific ontologies to capture knowledge of a particular enterprise (Missikoff et al. 2015). Some of the methodologies are very rigid and well detailed such as (Vrandecic et al. 2005) and (Kotis & Vouros 2006). Others are more liberal and simple with the aim to involve domain experts, who are not experts in ontologies (Auer 2006).

1.1.3 Conceptual Modeling

Conceptual models describe formal aspects of physical and social world, for the purpose of communication and understanding (Mylopoulos 1992). Conceptual modeling is regarded as a related discipline to requirements engineering (Shanks & Darke 1997) and software engineering (Moody 2005) as conceptual models are used to capture user requirements and build information systems satisfying those requirements. Currently, conceptual modeling goes beyond requirements engineering. The purpose of the conceptual modeling task is to represent phenomena in a domain of choice. It is able to represent both: static (such as concepts, properties and relations) and dynamic phenomena (such as processes and events) (Commentary et al. 2002). Gemino and Wand describe the conceptual modeling task as “a process whereby individuals reason and communicate about a domain in order to improve their common understanding of it” (Gemino & Wand 2004). Thus, different types of conceptual models (data models, requirements models, process models, etc) are used by enterprises for prescribing a certain reality in a company. In (Mehmood et al. 2009) conceptual models are associated with the following objectives:

- Meeting user requirements
- Formally representing observed reality
- Serving as basis for implementation and evaluation of information systems

In this PhD project two types of conceptual modeling languages will be used: business process modeling and goal modeling. We have opted for business process modeling as the first demonstration of our work as according to Davies and colleagues, process modeling was the primary reason to engage in conceptual modeling (Davies et al. 2006). Moreover modeling business processes has become a practice in many organizations (Rosemann 2006), (Pittke et al. 2013). However, despite the importance and wide spread use of those conceptual models, the possibility to retrieve and reuse the captured knowledge is still limited (Lin et al. 2006).

Where process models are used to describe the processes within a particular enterprise, goal modeling covers a completely different but yet very important aspect: it captures the motivation and strategy behind organizational practices (Yu et al. 2006). Goal-oriented modeling became increasingly popular among requirements engineering techniques (Matulevičius et al. 2007). It is primarily used to support early IS development stages as it models the rationale and scope of the IS and helps in resolving conflicts and making sure that all the stakeholders are heard by the development team (Matulevicius et al. 2015). Goal models are capable of representing functional and non-functional requirements (Santos et al. 2010), and shift from traditional modeling focus on *what* and *how* to innovative *who* and *why* (who are the actors, what they want to achieve and what is their motivation) (Moody et al. 2010). Moreover, by utilizing goal modeling, organizations can express their choices behind multiple alternatives, and investigate other possibilities (Jonkers et al. 2004).

Given such a great importance of both conceptual modeling languages, and their different points of focus, we believe that choosing those two options adds rigor to the demonstration of our approach. If it works for such diverse modeling types, then we can claim its generality. Demonstrating it for all (types of) conceptual modeling languages is obviously not possible within the scope of a doctoral thesis.

1.1.3.1 Business process modeling

A basic definition of business process is provided by Hammer, who defines it as “a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer” (Hammer 1990). Aguilar – Saven adds an “enterprise flavor” to the previous definition: “Business process is a partially ordered set of Enterprise Activities which can be executed to realize a given objective of an enterprise or a part of an enterprise to achieve some desired end-result” (Aguilar-Savén 2004). By aggregating both definitions, business process model receives an input, and generates an output by applying a partially ordered set of activities to satisfy business needs of an enterprise. (Lin et al. 2002) identified five essential elements in a business process: (1) a business process has a customer, (2) it constitutes of activities, (3) those activities aim at creating value to the customer, (4) those activities are executed either by people or by agents and (5) a business process often involves several units of the enterprise.

There are a plethora of research efforts concerning business processes modeling. (Aguilar-Savén 2004) perform a review of business process modeling literature. The author concludes that business processes are modeled for three main purposes: 1) to learn about the process, 2) to make decisions about the process and 3) to develop business process software. Based on which need is addressed, the type of process modeling technique is selected. The same author classifies business processes into “core” and “supportive”. “Core” is a primary business process which is initiated externally to the enterprise. The “supportive” business process is secondary and it facilitates the execution of the core process. The “supportive” business process can serve as management process which controls overall strategy and objectives of the enterprise.

Many process modeling approaches are proposed in the literature. Kueng et al (1996) classify those process modeling approaches into four main categories:

1. **Activity-oriented approach:** it primarily concentrates on activities (sometimes referred to as tasks). Using this approach, a business process is presented as a set of ordered activities. This representation is suitable for the high-level view on the process, but it might underestimate the true complexity of the work. It disregards the information flow and the involved enterprise units.
2. **Object-oriented approach:** this presents encapsulation, specialization, composition, and other aspect of object – oriented methods (Booch 1994). However, according to the author, this approach is not adequate for business process modeling.
3. **Role-oriented approach:** this modeling approach rotates around “role” concept, which is defined as a position played in a process by an individual, team, or unit. This is a rather broad definition, as it can imply a complete job description, such as school teacher, or a part of the job, or it can be limited to a specific task. The advantage of this modeling approach is the possibility to group activities and assign them to a particular actor. However, this approach does not allow expressing an intricate sequencing logic.
4. **Speech-act oriented approach:** the main concept behind this approach is “ActionWorkflow Loop”: every communication process (actionWorkflow) incorporates customer and performer. The communication process itself constitutes of four phases (loop): proposal, agreement, performance, satisfaction. Even though the idea itself is innovative, the approach is not practical in analyzing existing processes or in creating new ones.

More recently, a new paradigm of process modeling has emerged: declarative process modeling. The difference between imperative (traditional) and declarative process modeling is rooted in computer programming (Pichler et al. 2011). *Imperative process modeling* focuses on a “inside-to-outside” approach; it prescribes how exactly the work needs to be executed. All the possible alternatives need to be specified in advance before the execution starts. *Declarative process modeling* follows the “outside-to-inside” approach. It only describes the essential characteristics, and does not prescribe the precise execution procedure.

BPMN

The Business Process Modeling Notation (Object Management Group (OMG) 2008) or Business Process Model And Notation (OMG 2011) (BPMN) was developed by the Business Process Management Initiative (BPMP) Notation Working Group after two years of effort. The main purpose was to develop a notation which is understandable by both business users and developers. BPMN allows various departments within an enterprise to understand their processes, and enables an enterprise as a whole to communicate its processes in a standardized manner. BPMN is designed to cover various types of models and as a result, it enables the creation of end to end business process. As described in (OMG 2006a), there are three basic types of sub models within BPMN 2.0:

1. Processes, which include private non executable business process, private executable business process, and public process. *Private* processes are internal to the organization, while *public* process represent interaction between the organization and external participants.
2. Choreographies: while a regular process is contained in a pool (see Appendix A), choreography exists between pools and represents a definition of an expected behavior.
3. Collaborations: represents interaction between two or more business entities by means of public processes

To learn more about BPMN, the reader can refer to (Silver 2009). BPMN is a relatively young notation dating to 2004. Nevertheless, it underwent several updates. The latest updates (at the time of submitting this thesis) can be found at (Allweyer 2016).

BPMN is capable of conveying a wide variety of information to a broad audience. White identifies two main types of business process diagrams: Collaborative (public) B2B process, and Internal (private) business process (White 2004). The collaborative B2B process captures an interaction between two or more business entities in a neutral way. This process does not take a point of view of any of the organizations. It simply depicts the sequence of activities, and the communication among those organizations. The second type,

Internal business process, acts from the perspective of a particular organization, and depicts its interactions with others (external organizations).

For an overview of BPMN constructs the reader is referred to Appendix A.

1.1.3.2 Goal modeling

Zave and Jackson define a goal as an objective the system under consideration should achieve (Zave & Jackson 1997). Goals cover different types of objectives: functional and non-functional. Functional are associated with the services to be provided, while non-functional deal with the quality of service such as performance and accuracy (Van Lamsweerde 2001).

There are many reasons why goals are important in requirements engineering. Here are some of them:

- Goals provide rationale for requirements, thereby making it easier to explain to stakeholders (Mostow 1985)
- Goal refinement assists in structuring complex requirements documents thereby increasing readability (Van Lamsweerde 2001)
- Modeling goals explicitly helps in detecting conflicts in requirements (van Lamsweerde et al. 1998)

Goal models represent business objectives for stakeholders of an enterprise. Goal modeling is a promising and important methodology for eliciting requirements (Matulevicius et al. 2015; Shibaoka et al. 2007)

There is an established connection in the literature between goal modeling and business process modeling. Here are a few examples: Koliadis et al propose a framework which allows translating changes occurring in one model type (goal or process model) to the other model type (Koliadis et al. 2006). Decreus and Poels automatically derive BPMN models from requirements models which incorporate goal models (Decreus & Poels 2011). Finally, (Santos et al. 2010) proposes applying variability analyses on BPMN models using a goal oriented approach.

I* (I-star)

I* is an agent and goal-oriented modeling framework which analyzes intentional relationships among social actors (Yu et al. 2011). It is one of the most widespread goal-modeling approaches (Franch 2010) Unlike traditional systems and modeling methods which abstract from the social aspect, i* modeling revolves around social actors. Those actors are perceived as intentional; they have goals, beliefs, and commitments. The i* framework raised interest in socially-aware approaches to systems modeling, and led to several extensions and adaptations (Yu 2009). I* is mainly applied in requirements

engineering (especially for early requirements). Additionally, it is used in enterprise architecture, business process redesign, information security and privacy, digital asset protection, and software development among others (Yu et al. 2013). The main constructs of the i* notation are actor, goal, task, resource, and softgoal. The relationships are modeled through Strategic Dependency (SD) and Strategic Rationale (SR) models.

According to Franch, the i* framework reached a high level of maturity from an academic perspective, which resulted in a significant body of knowledge available through literature. Additionally, there are workshops, tutorial and scientific conferences for i*, and an i* wiki which offers explanation of the modeling language and supplies the reader with the initial literature. Despite all the research efforts, i* adoption by practitioners is unfortunately very limited (Franch 2010). For a detailed description of the i* notation the reader is referred to (Yu et al. 2011).

An overview of i* constructs is provided in Appendix B.

1.2 Conceptual Model vs Ontologies

Within the conceptual modelling research community there is no clear agreement with respect to the distinction between ontology and conceptual model. Some researcher consider ontologies as disguised conceptual models (Winter 2001). The literature presents several attempts to differentiate between ontologies and conceptual models which should help to set the boundaries for both. The confusion in literature between conceptual models and ontologies also has its origin in the fact that ontologies can support the conceptual modeling process. More details on how ontologies contribute to the semantic alignment of conceptual models are presented in Chapter 2.

A distinction that has been regularly mentioned is based on whether the model or ontology is descriptive or prescriptive in nature. A conceptual model that is developed within an enterprise forms the template according to which a particular enterprise system is developed (Aßmann & Zschaler 2006). For instance, a business process model prescribes the business process that is implemented by a process-aware information system. In contrast, ontologies are descriptive models, and therefore follow an open-world assumption (Horrocks et al. 2003) (anything which is not stated explicitly is unknown), rather than a closed-world as conceptual models. Ontologies portrait reality but reality is not constructed from them. If an ontology is used in a prescriptive manner, then it is better referred to as specification model (Seidewitz 2003). Most ontologies correspond to structural models whereas conceptual models can give a behavioral view in addition to the structural view. The second distinction proposed by the same author is related to sharedness (Aßmann & Zschaler 2006). An ontology is shared between different enterprises or between various

units and projects within one enterprise. A conceptual model does not need to be shared and can be developed and used by a small set of people within a specific context or project. The notion of sharedness is relative and therefore less useful to make a clear distinction between ontologies and conceptual models.

Within the context of this thesis we adopt the distinction between ontologies and conceptual models proposed by Fonseca and Martin (2007). The concept of *Ontology* described in this thesis is equivalent to the concept of *Computational ontology* in (Fonseca & Martin 2007), which implies an ontology used to build information systems (IS) within the Ontology-Driven IS development approach described by (Guarino 1998). The term *Conceptual model* appearing in this thesis, corresponds to Fonseca and Martin's *Conceptual schema*. Conceptual schema represents the results of the modeling process, merely a set of diagrams constructed in a particular modeling language to express data structures of an application to be developed. In their definition, Fonseca and Martin focus on the data models, however, this thesis incorporates different kinds of models, namely process and goal models. Hence, we borrow their definition and extend it with other kinds of models, not limited to data models. To avoid confusion, throughout this thesis we opt for using the term enterprise-specific ontology ESO (or simply, ontology), and conceptual modeling.

(Fonseca & Martin 2007) differentiates between ontologies and conceptual models based on the *objectives* they aim to satisfy, and the *objects* they incorporate. Those differences are summarized in Figure 2 below. With respect to the objectives:

- **Ontology** aims to offer explanation and information integration based on assumptions about invariant conditions defining the domain of interest. Ontologies supply the users of the IS with common assumptions about the whole, within which the facts about IS arise. Ontology aims to explain the domain by describing it as a coherent whole. It might support the development of an IS, or be used by an IS, but it is not entirely coupled to a specific IS. Ontology is able to serve a broad range of purposes and it supports the modeler during model creation and analyses. An ontology offers a common reference to which various artefacts related to particular projects within the enterprise are connected.
- **The Conceptual model's** objective is to classify the observed facts, and provide measurement (the linking of general categories with particulars) for the objects along dimensions of possible variation, and within the context of assumptions supplied by the corresponding ontology. Hence, models focus on linking general ontological categories with particular observations within the IS. Conceptual models are usually constructed with a specific IS in mind. Conceptual model aims to structure a set of

instances to facilitate querying the IS. Models from different projects within an enterprise are linked to one enterprise-specific ontology.

With respect to the incorporated objects:

- **Ontology** lays focus on reality and the real world. It represents the general and assumed categories defining the domain of interest. Ontology captures a domain which is intended to be shared by multiple projects within the enterprise. It encapsulates concepts and relationships which are meaningful within a context of a particular enterprise, but are not captured by the models. An ontology has a broader scope as it incorporates more than a conceptual model is able to cover. The ontology engineer is free to incorporate extra information to facilitate understanding of the concepts presented in the ontology.
- **Conceptual model** aims to link the ontology with the “facts”. It establishes the relationship between categories presented in the ontology and the range of possible variations of facts related to those ontological categories. For example, if an ontology contains a category of “Automobile”, a conceptual model will provide a machinery to link the category of automobile to a specific auto, owned by a particular person. The conceptual model represents a particular enterprise application, and the modeler is restricted by describing terms which are a part of the IS. Within this research project, even though all models are linked to the ontology, they are not restricted by the terminology used within the ontology. The modelers are free in selecting the terminology which suites the project for which the model is created.

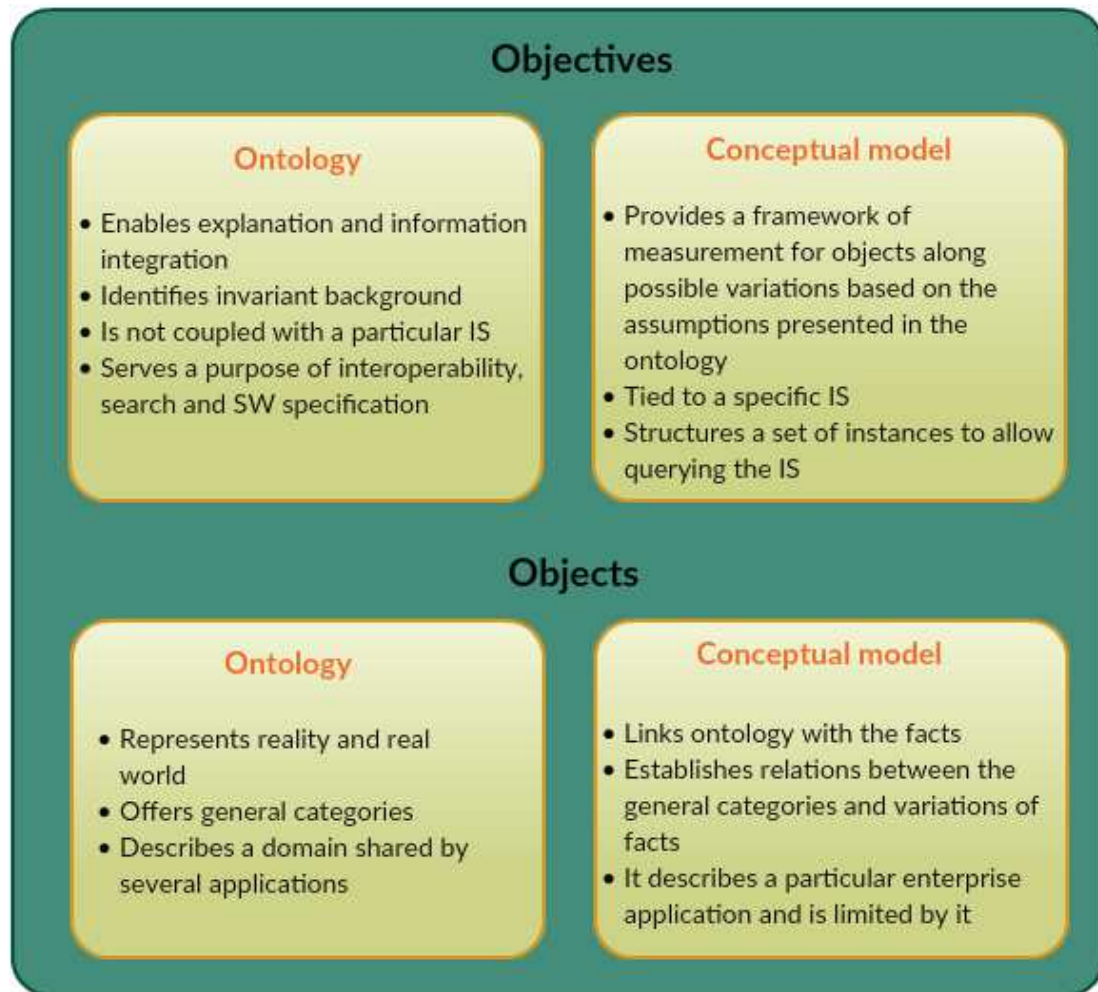


Figure 2: Differences between ontology and conceptual model taken from (Fonseca & Martin 2007)

1.3 Problem Description

Within an enterprise various types of conceptual models are created. Every model captures a particular concern, such as representing processes, intentions and goals, data, etc. All those models are represented in different modeling languages and created by different modelers. This diversity has a negative impact on the semantic alignment of those models. If an enterprise dedicates effort for model creation, it is important that those models are usable within different aspects of the enterprise. Models represented by different modeling languages are naturally lacking semantic alignment as every modeling language uses its own set of constructs, and mapping corresponding constructs manually can be a tedious task. Without additional effort for aligning those models, they will be hard to integrate and perform search. Additionally, models lacking semantic alignment will result in implementation of non-interoperable software systems. Models created by different people suffer label inconsistency and other signs of semantic heterogeneity as people tend to use different terminology to describe the same term. Furthermore, they can use different modeling constructs within one modeling language to model the same semantic concept.

Consequently, enterprise's knowledge base piles a big amount of semantically heterogynous models which are challenging to integrate and reuse.

This research work aims to improve semantic alignment of conceptual models created within one enterprise. This includes models of the same type created using the same modeling language, models of the same type created using different modeling languages, and models of different types. Within the context of this thesis, *Semantic Alignment* of conceptual models implies that overlapping elements of those models are consistent on the model and the meta-model levels. On the model level, model elements can be annotated with ESO concepts. Overlapping model elements are annotated with the same ESO concept, which ensures consistency of model elements. On the meta-model level, corresponding modeling elements are represented by equivalent modeling constructs in different modeling languages. The meta-model level is only applicable to models created in different modeling languages. Models lacking semantic alignment cause problems in communication among stakeholders, result in creation of non-interoperable software systems, and in additional effort and money wasted by the enterprise on integrating its different models, or systems built based on those models.

The following issues may arise when models are not aligned semantically:

- **Synonyms in labels of modeling elements carrying the same semantics.** Different people tend to use different words to depict the same concept as this depends on personal background and profession specific jargon. Models within the enterprise are created by different stakeholders working within different occupations, which can result in using occupation specific jargon. For example, a medical specialist may use “script” while a person without medical background will use the word “prescription” to depict the same meaning. Similarly, one modeler can use a word “client”, and another will use the word “customer”. Those words have the same meaning which is recognized by humans, but not by machines.
- **Abbreviations used in labelling modeling elements.** Humans tend to use abbreviations which are not recognised by machines even though they sound familiar to humans, such as using “dr” for “doctor”, “dept” for “department” and “h” for “hour”. Some abbreviations are not even recognized by the majority of people. For example, a stakeholder with a military background can use a label “NVS”, and other stakeholders without military background will not understand that it stands for “Night Vision System”. Other type of abbreviations can make a particular term ambiguous, such as using “report” to depict “business proposal report”. This is clear only in a very limited context. On a broader view “report” can stand for “research report”, “technical report” or any other type of report. Hence, using “report” is not precise and causes interoperability problems.

- **Inability to reuse knowledge captured in existing models.** Various conceptual models capture different aspects of the enterprise. Nevertheless, some models intersect and capture redundant information. This is particularly prominent in models of the same type representing overlapping concerns in different modeling languages, such as creating business process models using BPMN and UML activity diagrams. But this situation may also expand to models of different types; for example, process models may overlap with goal models while modeling a series of tasks to be executed to achieve a particular goal.
- **Inability to reuse general knowledge of the domain.** For example, if a Customer acquired a Loan from a Bank, then this relationship between the Customer and the Bank might be useful within the context of an enterprise operating in a financial domain. Some models might capture this relationship explicitly, while others will not. Independently, maintaining such a relationship as a common knowledge within the enterprise is important. It is best to define such a relationship at a higher level, not tight to a particular model. As a result, information about the relationship can be access without the need to search for models capturing it.

A possible solution for this model alignment problem is providing modelers with a shared vocabulary formalized in an ontology (Bera et al. 2011; Francescomarino & Tonella 2009). Enterprise-specific ontologies are gaining importance in the context of Data Governance and knowledge representation (Bera et al. 2011). Supporting tools, such as IBM InfoSphere¹ or Collibra Enterprise Glossary² allow enterprises to specify their own enterprise glossary/ontology. Such an enterprise-specific ontology, once available, can subsequently be deployed to help enterprise modelers in creating compatible, conceptual models, such as requirements, data or process models. If an enterprise decides to formalise its knowledge for reuse, it will be faced with various challenges regarding updating, retrieving and querying of the formalised knowledge. Therefore, developing effective practices in this regard is crucial. Another challenge is actually using this knowledge in model creation. How to select the right concepts? Where to look for them? What to do if the right concept is not found?

However, even if such an ontology is available, it might not be used by modelers during the modeling process. There are many reasons for such an underutilization of enterprise knowledge. For example, an interface to the ontology is too complex for a modeler without ontology training, and the absence of supporting tools which assist the modeler to navigate through the ontology.

¹ <http://www-01.ibm.com/software/data/infosphere/>

² <http://www.collibra.com/>

Another challenge surrounding ESO is how to maintain such an ontology to remain relevant to the enterprise it was created for. An enterprise is an evolving entity which occasionally undergoes changes. Completely new and innovative concepts are emerging, other concepts are becoming obsolete. It is essential that the ontology formalizing knowledge of an enterprise is also updated accordingly. Knowledge which became obsolete has to be removed, while new pieces of knowledge are added. The ontology is created to be actively utilized within the enterprise, and if it is outdated, it cannot be of a great benefit anymore.

To summarize, three research problems are highlighted to be addressed in this thesis:

- Semantic alignment of different types of conceptual models created within an enterprise
- Utilization of ESO during the modeling process
- Keeping ESO up to date with the evolving nature of the enterprise

1.4 Research Goals

The research work presented in this thesis is a part of a bigger research project with the overall goal of establishing and demonstrating a symbiotic relationship between ontology engineering and conceptual modeling. This PhD project addresses the issues related to the conceptual modeling part, while the community-based ontology engineering part will be explored as part of another project. Hence, this PhD aims to establish requirements for a framework which guides modelers through creation of semantically aligned conceptual models, and facilitates ESO maintenance and usage. This framework must operate independently of a particular conceptual modeling type or language to cover all the needs of an enterprise. After establishing the general requirements and translating these requirements to a framework with different cycles and phases, we explore in detail the parts of the framework focusing on the conceptual modeling efforts. Next, those parts (phases) are instantiated and thoroughly evaluated. This results in the following goals for this PhD project:

1. **GOAL1:** Setting overall requirements for establishing the symbiotic relationship between conceptual modeling and ontology engineering, and establishing a framework which will deliver those requirements.
2. **GOAL2:** Providing more detailed explanation regarding those phases of the framework which are related to conceptual modelling. This includes pointing out what can be reused from the literature, and offering practical guidance regarding every phase.

3. **GOAL3:** Evaluating the phases responsible for the conceptual modeling side of the symbiosis by demonstrating the framework for BPMN process modeling language. This implies instantiating the relevant phases of the framework for BPMN, and implementing a tool in order to perform an experimental evaluation. This will represent the first step towards proving framework's generalizability. Next, the same phases will be instantiated for i* goal modeling.
4. **GOAL4:** Exploring the ability of the framework in aligning i* goal models with BPMN process models. After instantiating the relevant phases of the framework for BPMN and i*, we want to investigate the possibility of creating BPMN models which are semantically aligned with previously created and stored i* models. This will demonstrate the operation of the framework across modeling languages.

1.5 Methodology

This research contributes to solving a specific business problem, i.e. semantic alignment of conceptual model, and to create artefacts that can be directly used by enterprise workers. For this purpose the Design Science Research Methodology (DSRM) is perfectly suited (Hevner et al. 2004). The main goal of a design science project is to solve a practical problem. The problem at hand here is twofold: 1) improving semantic alignment of conceptual models, and (2) enriching, evolving and keeping the ESO up to date. Following (Wieringa & Heerkens 2006), a DSRM project is divided into *Engineering Cycles* (EC) that provide the necessary steps to design and evaluate an artefact, and associated *Research Cycles* (RC), responsible for resolving research related issues, such as establishing the state of the art for a specific problem, finding and adapting related techniques, etc. The main problem is thus decomposed into nested problems represented by ECs supported by RC(s).

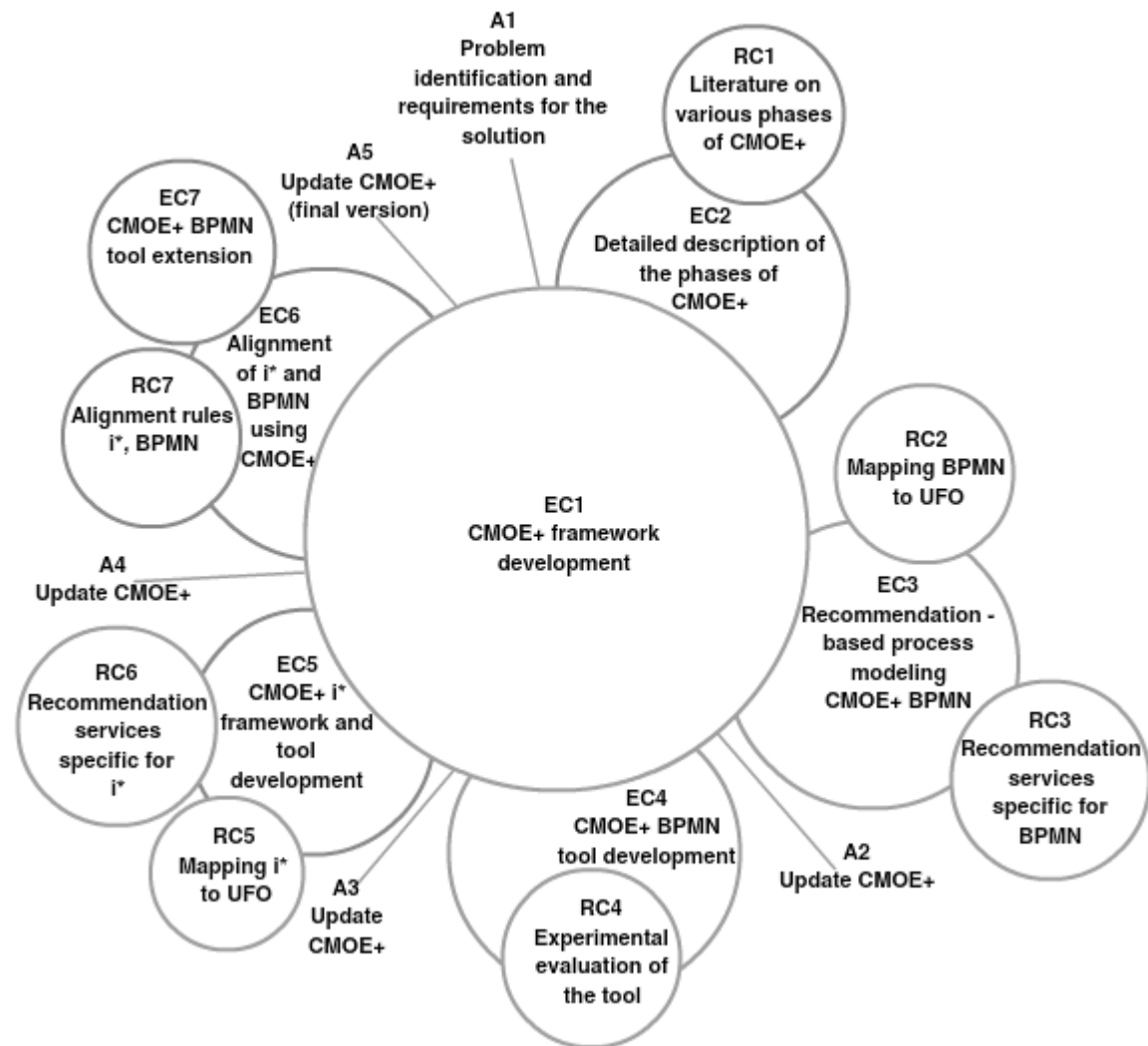


Figure 3: Regulative cycles of the PhD inspired by (Wieringa & Heerkens 2006)

This research project in Figure 3 commences with Problem identification and requirements specification activity **A1**. It focuses on analyzing the problem and defining requirements necessary for the solution. Based on the requirements obtained from A1 the main design science artefact, the Conceptual Modeling and Ontology Evolution framework (CMOE+), was developed in the subsequent engineering cycle **EC2**. Detailed description and ideas of EC2 were supported by RC1. This research cycle incorporated literature on various topics including semantic alignment of conceptual models, ontological analyses, community-based ontology engineering, and more. The CMOE+ framework consists of two cycles: an *Ontology Evolution cycle* which is responsible for ontology maintenance and amelioration, and a *Conceptual Modeling cycle* which facilitates creation of different conceptual models based on the ontology. The framework is explained in detail in Chapter 2. After designing the framework, it was applied to the process modeling (EC3 and EC4) and requirements engineering fields (EC5). EC1 was performed in iterations, and the first version was drafted in **EC2** based on the requirements resulting from A1.

EC3 represents the first instantiation of the CMOE+ framework. An ontology in the financial domain was selected as the starting point for the ontology engineering cycle of the framework. Process modeling using the BPMN modeling language was chosen for the conceptual modeling cycle. EC3 makes use of a broad knowledge base, therefore a thorough literature review was needed. In **RC2** we investigated the possibility of using various core ontologies, and how to map BPMN constructs to the selected core ontology. **RC3** is about finding the optimal combination of matching algorithms that will reduce to the minimum the effort required by the modeler to locate the right concept in the ontology. Some matching algorithms were reused, some specifically designed. Upon completion of EC3, the first modifications were made to the framework (**A2**). All findings of RC2 and RC3 were implemented in a tool (**EC4**) that allowed us to execute experiments with students and evaluate this first instantiation of the framework (**RC4**). Two experiments were conducted with the tool: one at the Vrije Universiteit Brussel, and one at Ghent University. Based on the findings, CMOE+ was enhanced again in **A3**.

To ensure that CMOE+ is suited for different modeling languages and is not biased to one type of conceptual models, the framework was tested in the goal modeling field using the i* modeling language. This resulted in implementing CMOE+ i* tool in **EC5**. The CMOE+ framework was updated accordingly in **A4**. After presenting the performance of CMOE+ for individual modeling languages, it was necessary to demonstrate that the framework is able to operate across different modeling languages. **EC6** establishes the alignment between goal models in i* and process models in BPMN. This is achieved with the help of **RC7** which explores the possibilities of various alignment rules. **EC7** extends the CMOE+ BPMN, implemented within EC4, to offer evidence on the alignment. After completing this last engineering cycle, CMOE+ framework was altered again (**A5**), which resulted in its final version.

1.6 Publications

This section presents the published and submitted papers written during the course of this PhD. the section gives a brief overview of the content of the paper, and the chapter(s) where it was used within this thesis.

- F. Gailly, S. Casteleyn, and N. Alkhaldi. (2013). On the Symbiosis between Enterprise Modeling and Ontology Engineering. *Proceedings of 32nd International Conference on Conceptual Modeling*. pp. 487 –494

This paper is our first publication and it gives an overview of the initial version of CMOE+ framework. This framework was modified and enhanced in iterations, with every instantiation and evaluation. Hence, this first paper does not include all the details, and uses a slightly different terminology. Some terminology was changed

with the advancement of this research; while presenting our achievements in conferences, we noticed the confusion caused by some terminology, and tried to adapt the terminology to achieve the common understanding with other researchers in the field. This paper is incorporated in Chapter 2.

- N. Alkhaldi, S. Casteleyn, and F. Gailly. (2014). Supporting Process Model Development with Enterprise-Specific Ontologies. *Proceedings of the 16th International Conference on Enterprise Information Systems*. pp. 236–248

This publication presents an enhanced version of CMOE+, and its instantiation for process modeling using BPMN with a basic demonstration of the instantiation process. This paper is not included in the thesis as it presents only the basic idea behind recommendation – based process modeling.

- Alkhaldi, N., Casteleyn, S. and Gailly, F. (2015). Enterprise-specific Ontology-driven Process Modeling. In J. Cordeiro, S. Hammoudi, L. Maciaszek, O. Camp, & J. Filipe (Eds.), *Lecture Notes in Business Information Processing*. pp. 472-488 Springer

This work provides a detailed overview of recommendation – based process modeling, which is the first instantiation of CMOE+ framework. This paper is a part of Chapter 2.

- F. Gailly, N. Alkhaldi, S. Casteleyn, and W. Verbeke. (2017). Recommendation-based Conceptual Modeling and Ontology Evolution Framework (CMOE+). *Business & Information Systems Engineering*

It presents an evaluation of CMOE+ for process modeling. It describes the developed process modeling tool, and the experiment conducted with university students to evaluate the feasibility of such a modeling tool. This paper is presented in Chapter 3.

1.7 Thesis Structure

This PhD thesis constitutes of five chapters. Chapters' division is inspired by the engineering cycles presented in Figure 3 (Section 1.5).

CHAPTER 1: Introduction

This Chapter defines relevant topics, methodologies, and concepts used within this thesis namely “ontology”, “ontology engineering” and “conceptual modeling” (Section 1.1). Further, it clarifies the difference between ontology and conceptual models within the context of this thesis in Section 1.2. Next, this chapter describes the problem to be addressed (Section 1.3) and states the goals to be accomplished (Section 1.4). Section 1.5 highlights Design Science Research Methodology, the

methodology followed within the course of this PhD. Section 1.6 lists the papers published and submitted during the years of PhD research. The final section, Section 1.7, presents the overview of this thesis.

CHAPTER 2: Recommendation-Based Conceptual Modeling and Ontology Evolution Framework

Chapter 2 starts with landscaping the existing research and classifies related literature in Section 2.1. Section 2.2 identifies the requirements for CMOE+. Next, Section 2.3 establishes the different phases of CMOE+, and provides a detailed explanation of conceptual modeling-related phases of the framework. Finally, Section 2.4 highlights the correspondence between the requirements and the various phases of CMOE+.

CHAPTER 3: Recommendation-Based Process Modeling

Chapter 3 applies CMOE+ to process modeling using BPMN. It illustrates the instantiation of various phases of CMOE+ (Section 3.2). Section 3.3 presents the tool which was developed to evaluate CMOE+. Finally, Section 3.4 presents an experiment that was conducted with students to evaluate the CMOE+ on process modeling. The focus of this experiment lays in evaluating user acceptance of the different modeling style.

CHAPTER 4: Aligning I* Models and BPMN Models Using the CMOE+ Framework

This chapter presents the application of CMOE+ framework in aligning goal models in i* with process models in BPMN. The first section offers an overview of existing research on the synergy between goal and process modeling. The following section elaborates on CMOE+i* framework and tool development (Section 4.2). Section 4.3 explains the view of this thesis on the alignment between i* and BPMN models, and the features added to CMOE+ BPMN method to accommodate the alignment. Next, Section 4.4 offers a demonstrative example featuring the alignment rules with a simplified emergency department case. Section 4.5, contributes to developing a better understanding of the CMOE+ tools generally. It presents a component diagram of the tool, highlighting the components which require adaptation while configuring a new modeling language with CMOE+.

CHAPTER 5: Conclusions and Future Research

This chapter iterates on the research results, summarizes the contributions, explains the limitations, and research efforts planned for the future.

2

Recommendation-Based Conceptual Modeling and Ontology Evolution Framework

This chapter elaborates on the Recommendation-Based Conceptual Modeling and Ontology Evolution framework (CMOE+). The first section provides an overview of existing literature on conceptual model alignment. Next, this chapter presents the requirements for the Recommendation-Based Conceptual Modeling and Ontology Evolution framework (CMOE+). Afterwards, an overview of different cycles and phases of CMOE+ is presented. Phases concerning conceptual modeling activities are elaborated in detail. Finally, this chapter summarizes the correspondence between the requirements of CMOE+ framework and its different phases.

2.1 Semantic Alignment of Conceptual Models

In modern day enterprise engineering, it is paramount that conceptual models are grounded in a well-defined, agreed-upon Enterprise Architecture that captures the essentials of the business, IT, and its evolution. Enterprise architectures typically contain different views (e.g. Business, Information, Process, Application, Technical) on the enterprise that are developed by distinct stakeholders with a different background and knowledge of the business. Consequently, the developed conceptual models that populate these views are hard to integrate. This lack of semantic alignment deteriorates the quality of the modeling landscape and the analyses performed on it (Jorg Becker, Breuker, et al. 2009).

Researchers have attempted to tackle this alignment problem fully or partially. Some researchers refer to this issue with a term of “consistency” (Mens et al. 2005), and others favour the term of “interoperability” (Lin & Strasunskas 2005). Throughout this thesis the term “semantic Alignment” will be used following the definition supplied in Section 1.3. Existing research works featuring “interoperability” and “consistency” are consulted during the literature review, and some of their contributions are partially reused.

One approach for solving integration and alignment issues between conceptual models is using Enterprise Architectures (Lankhorst 2005). Techniques like TOGAF (The Open Group n.d.), Zachman (Sowa, JA and Zachman 1992) and Archimate (The Open Group 2013) focus on describing different aspects of an enterprise in an integrated way. The actual level of integration realised between the perspectives of the enterprise architecture and the approach followed depends on the used framework. For instance, Archimate defines a new language for enterprise modeling which should be reused within the different perspectives. Zachman organizes and classifies the descriptive representation of the enterprise in a matrix where every cell is an artefact of the enterprise. TOGAF focuses more on the process of creating the different conceptual models and aligning them.

Another possible solution to the semantic alignment problem is using a shared terminology during the development of these different views (P. Bera et al. 2011). Such explicit formal representations, often materialized in the form of an ontology – in a business context called an enterprise-specific ontology - provide a myriad of advantages:

1. It is generally accepted that ontologies can be used to assist in communication between human agents, to achieve interoperability, or to improve the software engineering process (Uschold & Jasper 1999).
2. On an intra-organizational level, they ensure model re-usability, compatibility and interoperability, and form an excellent basis for supporting enterprise applications, such as Enterprise Resource Planning (ERP) systems, supply chain systems or business intelligence (BI) tools, for which they serve as common terminology.
3. On an inter-organizational level, ontologies facilitate interoperability, cooperation and integration by allowing formal mappings between, and alignment of separately developed conceptual models. While a wide range of more generic enterprise ontologies with a various intended use are available (Geerts & McCarthy 1999; Gordijn & J. M. Akkermans 2001; Uschold et al. 1998; Osterwalder & Pigneur 2002), they are often not immediately usable by a particular enterprise, as they lack enterprise-specific business concepts which enterprise-specific ontologies do provide, or do not offer a complete coverage of the business domain.

Existing ontology – based approaches for improving semantic alignment of conceptual models can be classified in two dimensions: First, approaches which enhance alignment by means of the modeling language vs approaches which affect the conceptual model itself. Second, approaches that improve model alignment after the model is created vs approaches enforcing semantic alignment while the model is being created by the modeler. This approach to literature classification is represented in Table 1.

Table 1: Existing approaches to enhancing semantic alignment of models using ontology

Modeling language / after model creation	Model / after model creation
(Opdahl et al. 2012) (Uschold et al. 1998) (Osterwalder & Pigneur 2002) (Gordijn & H. Akkermans 2001)	(Pittke et al. 2013) (Born et al. 2007) (Francescomarino & Tonella 2009) (Thomas & Fellmann M.A. 2009) (Si-Said Cherfi et al. 2013) (Di Martino et al. 2016)
Modeling language / during model creation	Model / during model creation
(Jörg Becker et al. 2009) (Pfeiffer 2007) (Geerts & McCarthy 1999) (Evermann & Wand 2005a) (Sonnenberg et al. 2011) (Rospocher et al. 2014)	CMOE+ (Jorg Becker, Delfmann, et al. 2009) (Delfmann 2009)

In Table 1, starting with the upper left corner, in (Opdahl et al. 2012) within the UEML (Unified Enterprise Modeling Language) project, the constructs of different conceptual modeling languages are mapped to an intermediate language, which has its origin in the Bunge Wand Weber ontology (Wand & Weber 1988). Next, these ontological mappings are used to enforce alignment between models. The Enterprise ontologies mentioned in the introduction (Uschold et al. 1998; Geerts & McCarthy 1999; Osterwalder & Pigneur 2002; Gordijn & H. Akkermans 2001) are mostly used to develop an enterprise modeling language which is immediately applied during the creation of the model. The work of Becker et al (Jorg Becker, Breuker, et al. 2009), which is based on the ideas of Pfeiffer (2007), uses a domain-specific modeling language to constrain modeling choices, aiming to avoid model variations and promote semantic alignment. Sonnenberg et al (2011) put forward a domain specific modeling language to support creation of more interoperable conceptual models specialized in economic events. Evermann & Wand (2005a) extends object – oriented languages with

ontological mappings to make it more suitable for conceptual modeling, while (Rospocher et al. 2014) offers ontological description of BPMN modeling language.

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), Di Francescomarino and Tonella (2009) and Si-Said et al (2013) 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 modeler to use naming conventions while adding 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 modeling domain, semantically enriched process models are not only used to align process models. They can also be used to automatically analyze business processes (Fill 2012; Fill 2011b; Becker et al. 2010) 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 (Fill 2011a; Thomas et al. 2009) or allow transforming a process model into a semantic business process (Martin Hepp et al. 2005; Cabral et al. 2009; Abramowicz et al. 2007b).

All previously described approaches for solving alignment and integration issues between conceptual models differ from the approach proposed in this work. First, we focus on using an enterprise-specific ontology and not generic enterprise ontologies that are shared by different enterprise. Second, we do not impose particular modeling language(s) but instead allow the use of generally accepted languages that captures different aspects of the business. Finally, the enterprise-specific ontology is used to provide suggestions to the modeler during model creation, and it evolves according to the specific needs of the enterprise. The fact that the enterprise-specific ontology is used throughout this research, does not exclude the possibility of model alignment across multiple enterprises. The goal of this particular thesis is to align models within one organization assuming its autonomy. However, this is only the first step. When all models within the enterprise are created in a semantically aligned manner, the enterprise can map its ESO to the ESO of its partner organizations. This will help significantly in aligning models of both enterprises, thereby proving useful on an inter-organizational level.

2.2 Requirements for CMOE+

As was mentioned in the previous chapter, it is a common practice in an enterprise that models are created by different stakeholders, with different backgrounds and using a different vocabulary set. As a result, the models are not semantically aligned, difficult to reuse, and need human intervention to be integrated. To resolve this issue, there is a need for a formal basis in which those different models can be grounded. The CMOE+ framework proposes using an enterprise – specific ontology to bring uniformity to all the models created within one enterprise. The main goal of CMOE+ is to produce models which are semantically aligned from the start, rather than increasing model alignment as an independent process. As CMOE+ relies on Enterprise-Specific Ontology (ESO) for model unification, another important goal is to maintain this ESO and ameliorate it as the enterprise evolves.

In order to understand the problem, specify the scope and the research questions, we cooperated with a contact person from Collibra³, and looked into the Flanders Research Information Space (FRIS) project. Both the company and the case study were very relevant to the problem at hand. The FRIS project (Debruyne et al. 2011) aims to collect and publish information on research entities such as researchers, research institutions and projects. This will reduce the administrative work of universities so that they do not need to report the same information in different formats. Currently FRIS offers some free services based on the mash-up of data on main entities and their relationships. The main entities in the FRIS ontology are Project Proposal, Project and Funding Program. FRIS has a diverse community of actors including high class actors such as minister of innovation, and middle class actors such as researchers and program managers. They all are given an opportunity to create and modify the FRIS ontology. To summarize, FRIS has one ontology which forms the basis for creation of different types of models and services (such as reporting) by different stakeholders. Additionally, the stakeholders have the possibility to request modification of ontological concepts. Those two aspects make this case very relevant as a starting point of this research work. It helps to understand the current state of affairs, and to specify requirements that we need for our solution.

Due to the circumstances, our involvement in this case was limited. A few meetings with the contact person from Collibra took place, and this helped in understanding community-based ontology evolution and its usage. Additionally, we participated in a workshop on establishing the process to be followed by FRIS participants to request changes to the ontology (ontology evolution). Being a part of the FRIS case allowed us to observe the ontology evolution methodology utilized within this case, and it enforced the importance of following an established methodology. Another significant contribution of the case was

³ <https://www.collibra.com/>

conducting an interview with the member of R&D department at the Vrije Universiteit Brussel, who was involved in the FRIS case. This interviewee informed us that his department is using their own independent system for storing facts about scientific output which needed to be integrated with the FRIS ontology. The R&D department had their information system and their established way of working. Despite the requirement to connect to the FRIS ontology, VUB had no desire to change the whole system accordingly, and would rather connect through an intermediary. This interview had a significant input towards the Conceptual Modeling cycle of CMOE+ (Section 2.3.2) as it demonstrated how different stakeholders utilize their own established systems, and do not want to change the way they work. They need a methodology to allow accessing the shared ontology without the need to adjust their working patterns.

Reading about the case, and participating in the meetings allowed to observe in practice the importance of an ontology in solving alignment problems. The FRIS case inspired us for most requirements related to the ontology evolution part. Moreover, requirement 1 (below) is directly derived from the case. The case included many different actors using the ontology for their particular purpose, which implies that the proposed solution is expected to be independent of a particular modeling language or paradigm to accommodate the variety of purposes the ontology is used for.

While studying the research problem, seven requirements were put forward for the solution:

REQ1: CMOE+ is a general framework not bound to a modeling language or ontology. CMOE+ must function in combination with different conceptual modeling languages, as it is a general and universal solution which is not bound to specific applications. This requirement is important as one enterprise typically utilizes several modeling languages and approaches. Binding the framework to only one modeling approach would significantly constraint its usage.

Proposed solution: This requirement can be achieved by maintaining the flexibility of the framework and making all its phases reconfigurable, not hardwired to a particular modeling paradigm. Providing guidelines on how to instantiate CMOE+ for different modeling languages also contributes to this requirement.

REQ2: CMOE+ must support to the modeler during the ESO-based conceptual modeling task. As the modeler is expected to create the model based on ESO ontology, CMOE+ must facilitate accessing the ontology for the modeler, and allow browsing the ESO in a convenient way. The ontology can be very extensive and performing a complete ontology lookup with every modeling element can be an overwhelming task. Hence, it is important to present ESO concepts to the modeler in an organized pattern.

Proposed solution: REQ2 can be achieved by incorporating mechanisms manipulating ESO concepts to bring the most relevant ones to the attention of the modeler. The

simplest organization would be presenting the ESO concepts alphabetically, which already offers more support than an unstructured presentation. However, CMOE+ goes beyond an alphabetical structure, and arranges ESO concepts based on relevance.

REQ3: CMOE+ must ensure creation of models with an appropriate quality level. Every organization employs its own quality standards, and if CMOE+ is to become organizational practice, it needs to ensure that those quality standards are satisfied.

Proposed solution: CMOE+ should incorporate a quality management framework which can be filled in according to the organizational standards. If quality requirements are not satisfied, the created models need to be re-engineered, and cannot be used in further phases of CMOE+ in their current state.

REQ4: CMOE+ must provide means to capture any update on the knowledge of the organization. An enterprise is an evolving entity; it is constantly changing; new innovations appear, some aspects are becoming outdated. This needs to be reflected in the ontology as the ontology formalizes the knowledge of the organization.

Proposed solution: CMOE+ aims to achieve this ontology update dynamically by means of conceptual models created within the enterprise. Following a more static approach, such as examining all artifacts of the enterprise on annual bases searching for updates, would create a big overhead. Therefore, capturing model feedback as potential content for ontology update forms a proactive approach for faster and easier ontology amelioration. The feedback is stored automatically while the model is being developed.

REQ5: CMOE+ must facilitate and guide community negotiation over the feedback from conceptual models. Not every feedback needs to be incorporated in the new ontology version. There is a rigorous selection process where the stakeholders of the enterprise decide what is relevant and what is not. If the feedback reflects only a localized change, then it is not broad enough to incorporate it in the ESO.

Proposed solution: In order to satisfy this requirement, CMOE+ incorporates a community guidance framework. Hence, several community members are required to review a particular feedback and acknowledge its relevance for the enterprise.

REQ6: CMOE+ must ensure that ESO remains up to date with the enterprise. In order to achieve that, CMOE+ promotes ontology amelioration with the processed feedback. The feedback selected by the community needs to be actually incorporated in the ontology. Otherwise the whole process is meaningless.

Proposed solution: CMOE+ framework needs to provide guidelines on how to incorporate those changes and how to manage all the related issues such as ontology versioning, and updating models already connected to the ontology.

REQ7: CMOE+ must maintain a semantic link between the modeling elements and the ontological concepts. This link represents a connection between the model and the ontology which allows achieving semantic alignment even if different wording is used. This type of semantic connection allows the modeler to exercise their freedom in labeling the modeling elements with labels different from the ones found in the ontology. Even though it is a best practice to reuse ontology terminology, some parts of the enterprise might have their own specific “jargon” which is not easily eliminated. Another important usage of this semantic connection is the ability to track affected models during ontology upgrade to a newer version. When particular concepts in the ontology are altered or deleted, the semantic link refers to the models where those altered or deleted concepts were utilized, allowing the modelers to take the necessary action.

Proposed solution: to satisfy this requirement, the implementation of CMOE+ must establish a link between all modeling elements in different models, and the ESO concepts they refer to.

2.3 Development of CMOE+

The previous section, Section 2.2, has established the requirements for CMOE+ which satisfies GOAL1 described in Section 1.4. This current section commences with presenting the two cycles and the phases constituting CMOE+ framework which is also a part of GOAL1. Next, phases concerning the conceptual modeling part of CMOE+ (Sections 2.3.1.1, 2.3.1.2, and all subsections of Section 2.3.2) are elaborated in detail to satisfy GOAL2.

The CMOE+ framework was built in iterations. First the high-level phases were developed based on the proposed requirements and inspired by the existing literature. Every time the framework was tested with a different modeling language, the gained experience has contributed to a better definition of its phases, and allowed making more concrete choices and decisions. Figure 4 below shows that CMOE+ consists of two cycles: an Ontology Evolution Cycle and a Conceptual Modeling Cycle. Both cycles are divided into several stages and frequently interact and intervene with each other. The flow of CMOE+ interchanges between phases of the two cycles, hence it is not possible to separate those cycles; they run in parallel.

The **Ontology Evolution Cycle** is responsible for obtaining the initial version of the ontology, maintaining it, and adapting it according to the changes in the enterprise. In the beginning of this cycle, the enterprise – specific and foundational ontologies are selected (or created). Those ontologies are formalized in an appropriate ontology representation format (RDF, OWL, XML, etc.), and stored for further use. Optimal ontology storage is vital for the operation of CMOE+ as it facilitates (or hinders) the access and retrieval of the ontology. During the modeling process, stored ontologies are accessed by the recommendation

services. Those services are specifically designed algorithms which search the ontology for suggestions for labels of modeling elements. Additionally, this cycle is responsible for gathering and evaluating feedback on the ESO. The evaluation is performed by a community of directly involved stakeholders. Finally, the feedback which passed the evaluation is properly incorporated in the ESO creating a new version. The ontology evolution cycle is responsible for maintaining various versions of the same ESO to ensure consistency among models created based on different versions.

The **Conceptual Modeling Cycle** facilitates conceptual model creation based on the ESO, and monitors the quality of the created model. Before the CMOE+ tool is at the modeler's disposal, ontological analyses of the modeling language(s) used for model creation need to be performed. The purpose of this step is to align the different modeling languages and the ESO. This alignment is utilized later by the recommendation services of the Ontology Evolution Cycle. The Conceptual Modeling Cycle is responsible for assisting the modeler in the model creation task by accessing and presenting the ESO ontology in an optimal way. The optimal representation is achieved with the help of the recommendation services (presented in Section 2.3.1.2). Finally, all created models are evaluated against quality standards of the enterprise. Conceptual models with poor quality are not entitled for feedback collection, and will be redesigned.

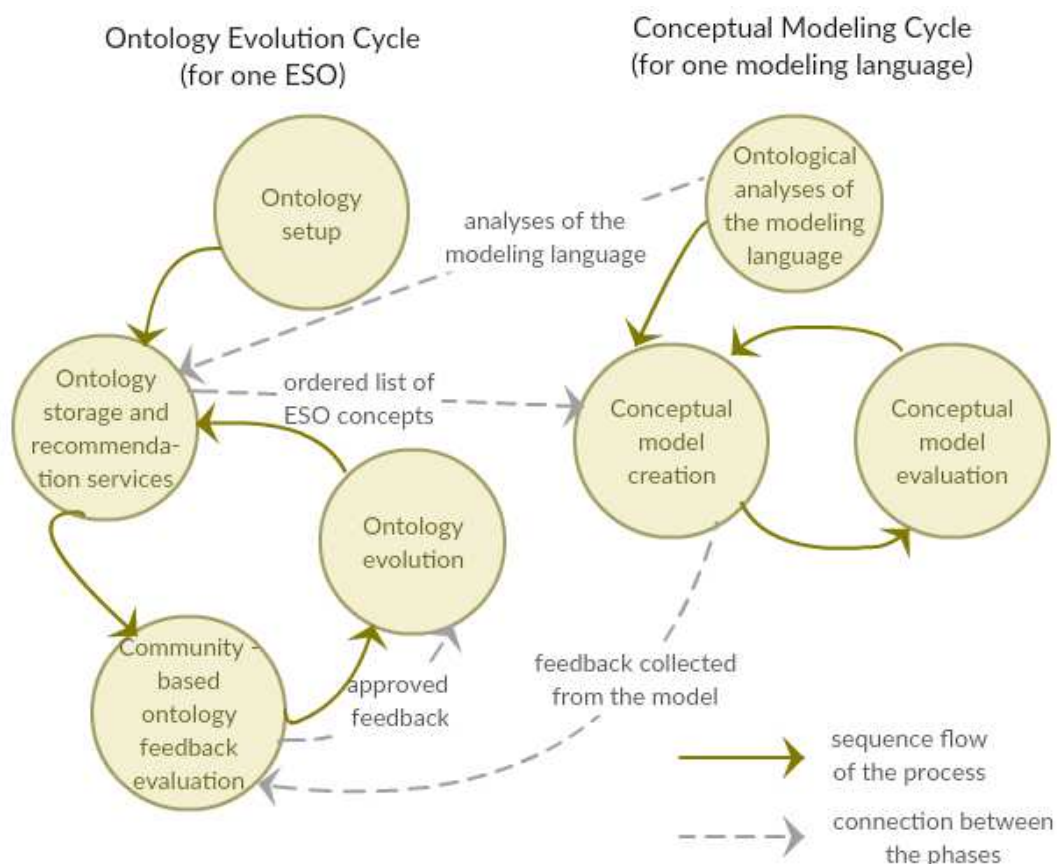


Figure 4: Different phases of CMOE+, and the interaction between them

As suggested by Hevner and colleagues in their Information Systems Research Framework (Hevner et al. 2004), every IS research is expected to utilize an existing research presented in the *knowledge base*. The knowledge base incorporates existing foundations and methodologies established by prior research. The knowledge base for our research is presented in Figure 5. This knowledge base featuring existing established literature consulted while developing various phases of CMOE+, and while instantiating it for different modeling languages. In the reminder of this section, we will describe each phase in more detail and point out how they are based on existing literature.

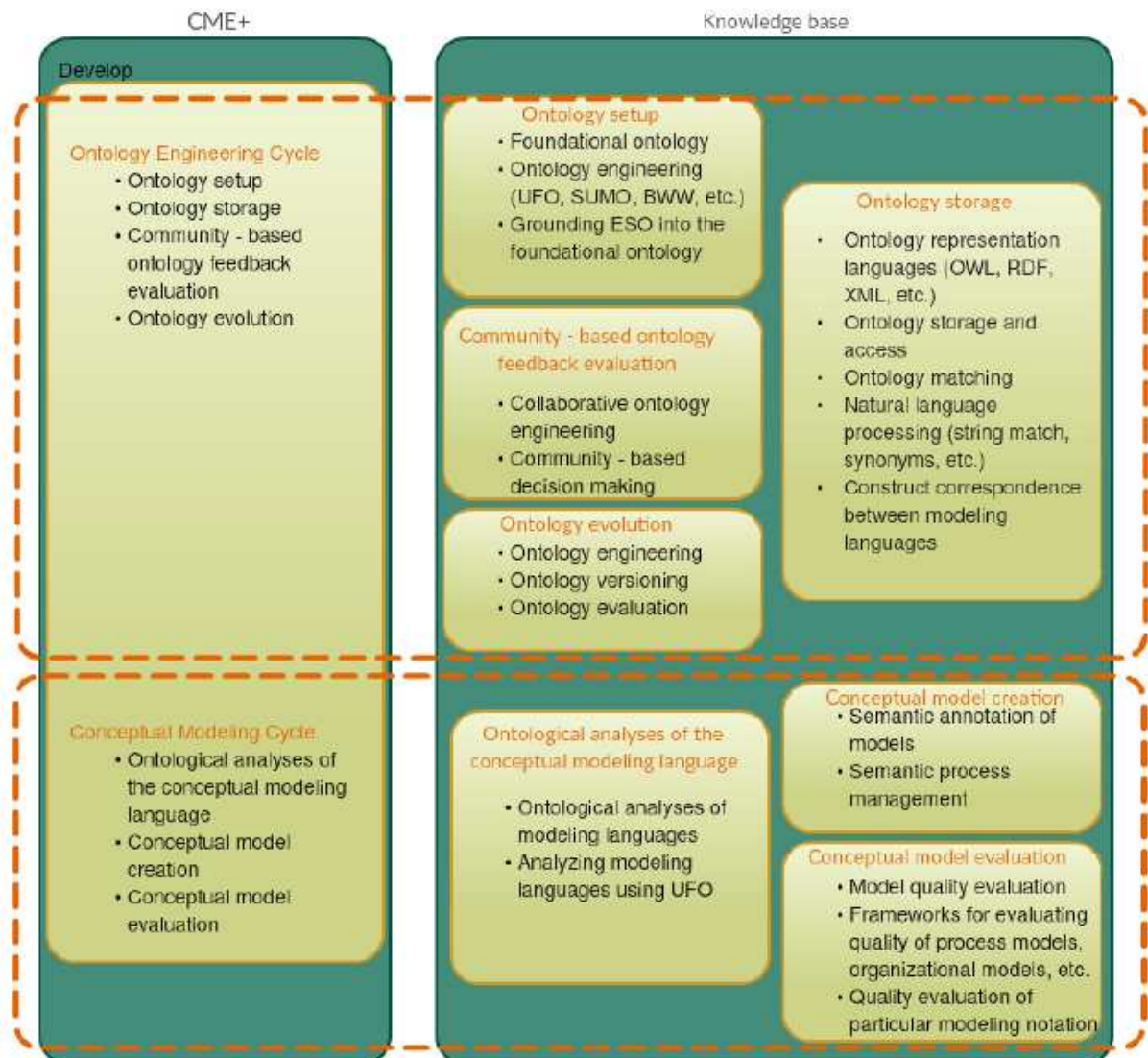


Figure 5: IS research framework followed during the development of CMOE+

2.3.1 Ontology Evolution Cycle:

This cycle constitutes of four phases: Ontology setup, Ontology storage, Community – based ontology feedback evaluation, and Ontology evolution

2.3.1.1 Ontology setup

During this phase the ontology engineer decides which foundational ontology, and (initial) ESO to use. Both ontologies are very important and their choice will significantly impact the whole framework. The selection of those ontologies, and the ontology representation format is free. However, there are several factors to consider. In the best case, an ESO is already available within the enterprise and can thus be used. If not available, an ESO can be created from existing business resources (e.g., glossaries, vocabularies, informal sources such as excel files of use case descriptions), or alternatively, an available domain ontology (domain of the enterprise) can be used as a starting point. The latter will subsequently evolve into the ESO as it is used within the iterations of CMOE+. The selected ESO may or may not be already mapped to a foundational ontology. Establishing those mappings is a very important task as the foundational ontology acts as a domain independent intermediary between the ESO and the selected modeling language (Section 2.3.2.1). Additionally, it facilitates operation of the recommendation services (Section 2.3.1.2).

A *foundational ontology* is an ontology that describes universally agreed upon, high level concepts and relations, such as objects, events, agents, etc. (Doerr et al. 2003). Hence, it adds well – founded semantics and forms the bases for information integration across different sources. Examples of a foundational ontology are SUMO (Niles & Pease 2001), UFO (Guizzardi & Wagner 2010a) and Bunge – Wand – Weber (Rosemann & Green 2002). CMOE+ does not restrict the selection of the foundational ontology. However, as it is the case for ESO, some choices can significantly reduce the required effort in this phase. It is best to choose a well-established and properly published and explained foundational ontology, as this will help in developing a good understanding of the ontology. In later phases of the framework, the same foundational ontology will be utilized to analyze the modeling language used for conceptual model creation. Consequently, it is advised to select a foundational ontology which has proven itself in ontological analyses of modeling languages. Even better, if the ontological analyses of the target modeling language(s) are already available.

In addition to ontology selection, the ontology setup phase is concerned with establishing the link (mapping) between the initial ESO (or domain ontology), and the foundational ontology. In other words: performing ontological analysis of the ESO. In literature, ontological analysis are mainly performed on modeling languages and the idea behind that is to add a sound foundation to the modeling concepts (Fettke & Loos 2003). If the initial ontology is not yet mapped to the foundational ontology, this will be the major challenge for the ontology engineer during this phase. A good mapping between the foundational ontology, the ESO, and the modeling language is a very important factor for the operation of recommendation services explained later in this chapter (Section 2.3.1.2). To evaluate the accuracy of grounding the initial ESO in the foundational ontology, the ontology engineer can utilize tools such as OntoClean (Guarino & Welty 2002a). OntoClean was developed to

assist in validating taxonomies and ontologies. Additionally, it is used to compare and integrate ontologies, and to assist in ontology creation. This methodology is based on general, formal notions which can be used with any ontology independently of its domain.

In order to accomplish this phase, there was a need to search for the optimal core ontology. After instantiating CMOE+ several times, for different modeling languages, we decided in this project to use the Unified Foundational Ontology (UFO) (Guizzardi 2012) as the foundational ontology in CMOE+ for three reasons: (1) the benefits of grounding domain ontologies in UFO are well motivated (Guizzardi & Wagner 2008), and several such UFO-grounded domain ontologies are available, e.g., (Barcellos et al. 2010) (2) UFO is specifically developed for the ontological analysis of modeling languages, which plays an important role in the next phase of CMOE+. (3) a variety of such analyses can be found in literature, e.g., analyzing BPMN using UFO core ontology (Guizzardi & Wagner 2011), analyzing UML (Guizzardi & Wagner 2010a). Despite this advice, which came by experience, the choice of the ontologies is still open and CMOE+ still supports any ontology without restriction.

Another practical part of this phase is analyzing the selected ESO using the foundational ontology. During the instantiation of CMOE+ we have used as ESO an existing domain ontology which we slightly enriched to take it to the level of an enterprise-specific ontology. The mappings between the ESO and the UFO are presented in Appendix D. It is advised to start with an enterprise-specific ontology if it can be obtained, as the results will be more precise. However, if obtaining the ESO is very costly, CMOE+ is capable of working with a domain ontology.

2.3.1.2 Ontological storage and recommendation services

This phase implies formalizing the ESO and the foundational ontology selected in the previous phase, and storing them in a way that facilitates access during the modeling process including search, retrieval, and reasoning. While going through further phases of CMOE+, there will be more ontologies that need to be formalized and stored. The selected storage and formalization techniques will have a great impact on the performance of CMOE+, therefore they must be selected with a careful consideration.

The first decision to be made is on how to formalize and represent the selected ontologies. Nowadays, various ontology representation languages are at the disposal of ontology engineers, such as OWL, RDF, XML, OIL, etc. It might be tempting to opt for the representation approach in which the selected ESO is represented. However, it is advised to examine various factors before making the decision, and it might be beneficial to rewrite the ESO in another representation language. The following factors need to be considered while selecting the ontology representation approach:

- It is best to opt for a generally accepted and widely used standard

- It is preferable if the selected ontology representation languages is compatible with several ontology engineering tool. This allows an ontology engineer to try another tool if one tool is not satisfying the requirements
- Ensure that the features required (such as reasoning, visualization, etc.) are supported by the selected representation language
- Verify whether the ontology representation languages is compatible with the storage media to be used

The second decision to be made during this phase is regarding the ontology storage medium. While selecting the appropriate storage media, factors such as performance and optimization need to be considered. Approaches to ontology storage include storing the ontologies locally, utilizing a web server for ontology storage, and using specific ontology databases.

Another important aspect to consider during this phase, is formalizing the rules to be used by the recommendation services (see next section) in the Rules Ontology (RulesO).

After extensive experience with the CMOE+ we have selected the Web Ontology Language (OWL)⁴ as ontology representation language for the following reasons: (1) it is a generally accepted standard; (2) Supported by most ontology engineering tools such as Protégé⁵; (3) supports various reasoning and retrieval features (such as punning) which helped in successful instantiation of CMOE+; (4) offers optimized storage media. Another choice we made is using the Stardog⁶ ontology database as ontology storage medium. While instantiating CMOE+, the authors have tried all the storage techniques mentioned above, and stardog database was proven to work the best. It supports OWL and the Java programming language. It offers excellent querying performance and allows handling big interconnected databases, which makes this technique appropriate for production.

In order to practically demonstrate this phase, several options of ontology storage were examined. Additionally, it was determined which types of different ontologies are required to be stored, and how to formalize them. For more information on the different ontologies, the reader is referred to Section 3.2.3.

Recommendation services

After the ontology is stored, it is accessed by the recommendation services while a conceptual model is being created. Every time the modeler places a modeling element on the canvas, several recommendation services cooperate in order to rank the ESO concepts and display the most relevant of them on the top of the suggestions list. This list is presented to the attention of the modeler only as an advice. There are no obligations for following the

⁴ <https://www.w3.org/TR/owl2-overview/>

⁵ <http://protege.stanford.edu/>

⁶ <http://www.stardog.com>

recommendations presented in the list. However, considering the variety of cooperating recommendation services, those suggestions have a good potential. Given the extensive amount of ESO concepts (which increases with time), relevance ranking of ESO concepts is a critical feature. Several approaches to narrowing down the set of available ontological concepts can be found in literature. Most researchers seem to heavily rely on linguistics and natural language processing of modeling elements' labels to suggest ontological concepts (Di Francescomarino & Tonella 2010; Thomas & Fellmann M.A. 2009). However, there are some research works which look beyond linguistics. Born et al in addition to label-based matching, consider the context of the modeling element, such as using the lifecycle of the domain object to suggest the next modeling activity (Born et al. 2007). Vazquez et al (Vazquez, Martinez, et al. 2013) derive two sets of suggestions: (1) Specific semantic annotation suggestions, which are based on a language – independent ontology (2) General semantic annotation suggestions, based on a domain ontology. Both sets of suggestions are derived based on the modeling construct instead of the label of the modeling element. To the best of our knowledge, the recommendation services presented in this paper are the most comprehensive and they incorporate very diverse aspects of matching. As it is mentioned in the beginning of this paragraph, some researchers focus exclusively on the natural language aspect (the label of the modeling element). And none of the works above are using ontological analyses to derive recommendations.

In CMOE+ three different recommendation services are used to rank ESO concepts. Those three services were selected because together they offer a comprehensive coverage as they exploit label similarity, and construct similarity of the modeling language. Additionally, the Rule-based recommendation service is very flexible and allows configuring rules which were not foreseen in advance. These mechanisms are partly inspired by ontology matching techniques (Euzenat & Shvaiko 2013), but are specifically focused to fit within CMOE+ framework, where the semantics of the modeling language can be exploited. Recommendation services are crucial for a successful adoption of CMOE+, and offer the following advantages:

- The ESO can be very extensive, and looking through all its concepts for every modeling element can be exhausting. Hence, displaying the most relevant concepts first can significantly reduce time and effort spent on browsing the ontology
- The suggestions list presents concepts that the modeler might not have thought of while creating the model
- It encourages the modeler from the start to utilize enterprise – specific terminology instead of his own jargon

Recommendation services operate not only based on the model under creation, but also can consider the modeling history (previously created models) of the enterprise. This feature was added to CMOE+ during the final iteration (Chapter 4)

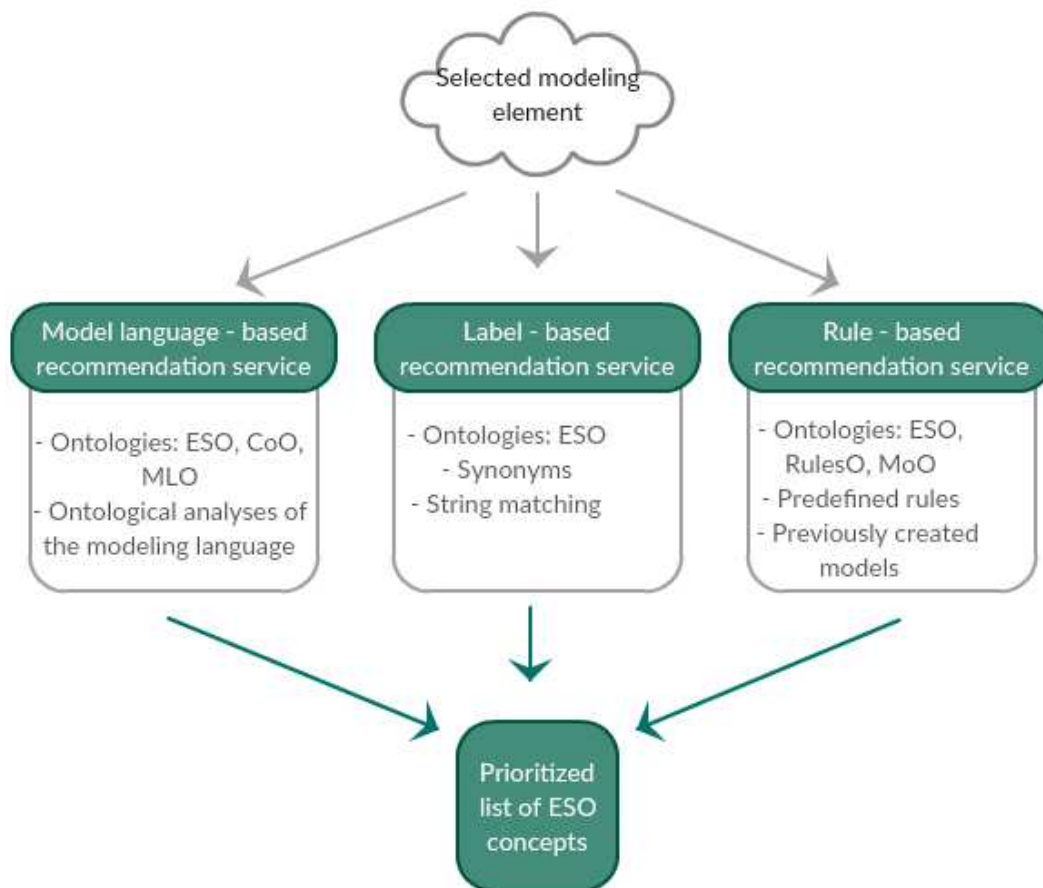


Figure 6: Recommendation services

Recommendation services operating within CMOE+ are illustrated in Figure 6. Every modeling element created by the modeler, is examined by three recommendation services: *Model language – based recommendation service* is concerned with the construct type of the modeling element. This recommendation service utilizes the Core (foundational) ontology (CoO) as an intermediary to connect the Model Language Ontology (MLO) to the ESO. *Label – based recommendation service* operates with the label of the modeling element and applies string match and synonym search to find the matching ESO concepts. Finally, *Rule – based recommendation service* examines the location of the modeling element within the model. Additionally, it can be configured to include other models within the model base of the enterprise formalized as Model Ontologies (MoO). It utilizes the ontology with predefined rules (RulesO) to find the matching ESO concept. The modeling process starts with initially predefined rules. However, those rules are configurable, and can be modified at any point of time.

Every recommendation service calculates a relevance score (between 0 and 1) for each ESO concept, with respect to the selecting modeling element. Subsequently, the overall relevance score is calculated using a weighted average of all individual scores. This corresponds to the formula below:

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

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

$S_{nb} W_{nb}$: the score and weight of neighborhood based match

It is important to note that the optimal weight and combination of recommendation services depends on the domain of the enterprise. For some domains a particular recommendation service might be more relevant and therefore assigned a higher weight. Within CMOE+ the final relevance score can be reconfigured to match preferences. Additionally, more experiments are needed to find the optimal score.

When the relevance score of every ESO concepts is calculated, the concepts are ranked and presented to the modeler. There are several possibilities to visually represent the ranking. One option is to display the most relevant concepts on the top of the suggestions list. In this case the modeler will see more relevant concepts from the beginning without the need to scroll throughout the whole ontology. The drawback of this method is that the order of the ESO concepts will change for every modeling element thereby confusing the modeler. An alternative of this display approach is to preserve the order of ESO concepts while changing the color, or otherwise highlighting the most relevant ESO concepts in the suggestions list. The benefit of this approach is that the order is fixed and will appear more familiar to the modeler. The disadvantage is the need to scroll through the whole list to view all the relevant concepts.

Offered recommendation services are very general and can operate with different ontologies and modeling languages. This gives CMOE+ its flexibility and facilitates its adaptation. Every recommendation service focuses on a different aspect of matching, thereby offering a comprehensive approach. Next, the different recommendation services will be described in more detail.

1) Model Language Recommendation Service

The key element of this recommendation service is ontological analyses of the conceptual modeling language used (described in Conceptual Modeling cycle). The ontologies utilized here are the ESO, the foundational (core) ontology (CoO), and the model language ontology (MLO). As presented in Figure 7, when the modeler selects a particular modeling element, this recommendation service operates by extracting ESO concepts which belong to the same type (of CoO) as the construct of the selected modeling element. Model language

recommendation service assigns a relevance score of 1 to ESO concepts of the same type, and the score of 0 otherwise. The success of this recommendation service depends on the selected foundational ontology; whether it is well-suited for ontological analyses, and offers an accurate correspondence between the ESO and the modeling language. Additionally, the success of this service largely depends on the ontology engineer performing the analyses, and their understanding of the selected foundational ontology.

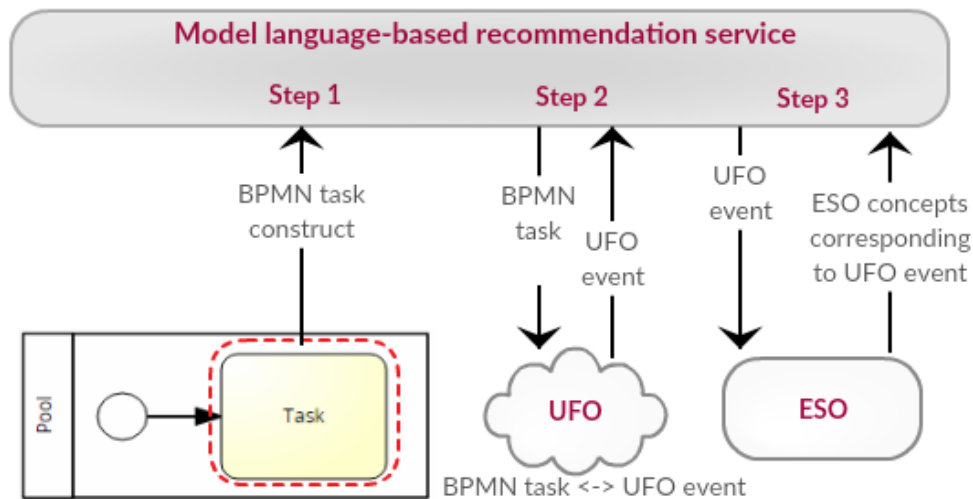


Figure 7: Model language-based recommendation service

2) Label – Based Recommendation Service

This recommendation service utilizes natural language processing techniques to analyze the label of the selected modeling element. As depicted in Figure 8, the analyses include two parts: (1) finding synonyms and (2) calculating lexical distance. First, the *lexical distance* is calculated between the label of the selected modeling element and every ESO concept. Considering Figure 8 below, the selected modeling element is labeled as “task”. This label is compared against every concept within ESO, and the lexical distance is calculated using a selected edit distance algorithm. Next, *synonyms* are extracted for every ESO concept, and the lexical distance is calculated for those synonyms. Revisiting Figure X, synonyms of each ESO concept are extracted using a pre-configured dictionary. The resulting synonyms are compared to the string “task” and the lexical distance is calculated. This allows finding ESO concepts such as “assignment”, “work”, “duty”, “effort”, etc., which are lexically similar to “task”, but would not be discovered using lexical distance calculating algorithm. Finally, for every ESO concept, the maximum lexical distance is used to determine the order of the concepts in the suggestions list. The score assigned to every ESO concept by the lexical distance algorithm is between 0 and 1. Among all the ontologies used within CMOE+, only the ESO is consulted at this stage.

Approaches similar to Label – based recommendation service are found in the literature. (Di Francescomarino & Tonella 2010) divides the label of the modeling element into verb and

object and looks for their match in the ontology. (Born et al. 2007) utilizes a combination of text and name matching techniques in their Name – based matching algorithm. (Niles & Pease 2013) offers a methodology to align WordNet (a linguistic database) to a foundational ontology.

This recommendation service is very flexible and allows configuring different natural language processing algorithms. After various instantiations of CMOE+, it was decided to use Jaro-Winkler distance (Winkler 1990), which is supported by the current version of CMOE+. For synonym retrieval, it is advised to use WordNet (Miller 1995) as it is a generally accepted, domain independent lexical database. However, it is possible to configure a domain-specific dictionary if required. And it is possible to manually add customized synonyms. The success of this recommendation service depends on the text matching algorithm used, and its suitability to the labeling style. It is preferable to use domain-specific dictionary, or extend WordNet with domain-specific synonyms. Otherwise, the desired results might not be achieved.



Figure 8: Label-based recommendation service

3) Rule – Based Recommendation Service

The Rule – based recommendation service depends on the matching rules which are defined and formalized in the Rules ontology (RulesO), and configured earlier during the Ontology storage phase. The rules can utilize different type of information that can have its origin in the used modeling language, the position of the selected element in the complete model, and the relationship among previously created models. Following are two examples of rules operating based on the location of the modeling element within the model, its relation to UFO, and annotation of other modeling elements in the same model. BPMN modeling language is used, as those rules are mostly language specific:

1. To create a BPMN message construct that results in transmitting a message between a task or event of a pool and another pool, the suggested ESO concepts (relevance

score 1) are corresponding to UFO relators mediating between ESO concepts selected to annotate the aforementioned BPMN pools.

2. For creation of a BPMN task construct, the suggested ESO concepts are most likely to be related through UFO material relations to the pool where the task is located. The suggestions can be either UFO quality types of the concept annotating that pool or UFO relators relating the ESO concept used to annotate the aforementioned BPMN pool to any other ESO concept.

As presented in Figure 9, after capturing a particular modeling element, the reasoner applies the rules and operates on different ontological files (including the ESO, RulesO formalizing the rules, Model Ontology (MoO) which represents the model under creation), and previously constructed models to create a repository with new annotations. Those annotations are displayed as suggestions to the modeler. A new rule can be added and configured at any time.

The rule-based recommendation service can benefit greatly from the existing research on construct correspondence among different modeling languages. Configuration of construct correspondence will allow the modeler to benefit from other models created within the enterprise. It sounds very obvious to consider existing models of the same type. For example, if a modeler is constructing a process model, he can benefit from the existing process models within the enterprise as they may contain tasks, or even complete sub processes of the process model under creation. However, not only models of the same type are perceived useful. Researchers are establishing connections between models of different types, in different modeling languages, representing various aspects of an enterprise. A few examples are: Endert et al map process models represented in BPMN to Agents and their attributes (Endert et al. 2007). Koliadis et al map BPMN constructs to i* (goal modeling language) constructs (Koliadis et al. 2006). Dijkman et al mapping BPMN models to Petri net (Dijkman et al. 2007). Those construct mappings can be formalized in the RulesO ontology in the beginning of this phase, and utilized by the Rule – based recommendation service.

As we opted for OWL as ontology representation language, we are using Semantic Web Rule Language (SWRL) to formalize the rules in RulesO file. SWRL is the preferred rule language to be used in combination with OWL.

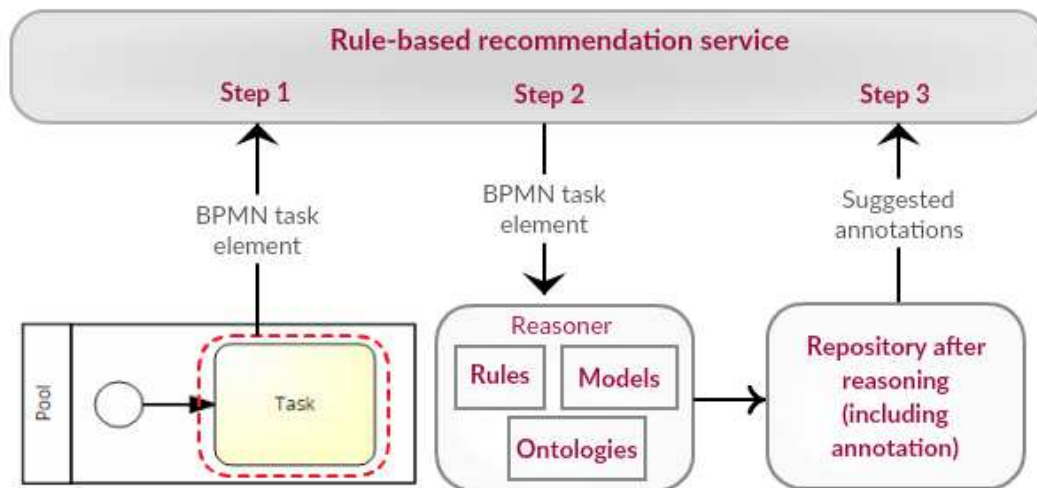


Figure 9: Rule-based recommendation service

2.3.1.3 Community – based ontology feedback evaluation

As it was mentioned before, CMOE+ framework aims to create semantically aligned conceptual models based on ESO, while simultaneously enriching the ESO using feedback captured from those models. In this thesis, only the direction concerning conceptual modeling task is addressed. Capturing feedback and utilizing it for ESO amelioration is a future work. This phase together with the Ontology evolution phase are only highlighted without diving into details. Hence, this Section together with Section 2.3.1.4 aim to satisfy research GOAL1.

This phase only starts if the resulted conceptual model satisfies the expected quality standards (when the Conceptual Modeling Cycle concludes), meaning that it correctly reflects the purpose it was designed for, and all the statements in the model are valid within its domain. This phase of the Ontology Evolution cycle, and the next phase (Ontology evolution) are based on the methodology of Zablith (2015). When the model is complete, two files are created and stored in the system for the purpose of extracting feedback. The first file represents the created model and the ESO concepts selected for the model annotation (annotation is explained in Section 2.3.2.2). It is stored as model ontology (MoO) represented in the selected ontology representation language. This file is processed in order to extract any possible feedback which is potentially useful and can be incorporated into the ESO. A possible feedback is a listing of the elements from the model that were not annotated by ESO concepts. As they were not annotated, the reason might be that there is no equivalent for them in the ESO. The second file is the log file which incorporates events occurred during model creation. The log includes suggestions derived by the recommendation services for every modeling element, suggestions accepted by the modeler, and model annotations deleted by the modeler. The log file helps to evaluate the work of the recommendation services, and gather feedback on deleted annotations. If an

annotation was deleted, it implies that the modeler found the appropriate ontological concept, but was not content with the name assigned to this concept in the ESO.

Both feedback files are first processed by the ontology engineer who extracts the candidate feedback which is potentially useful for updating the ESO. This feedback is made available for a selected group of stakeholders (community), and is subject to discussion, until finally a consensus is reached whether or not the proposed change(s) should be included (such as new concepts added) in the new version of the ontology. The community will discuss the proposed concepts such as their usage, definition and usefulness within the enterprise.

Within the ontology evolution cycle of (Zablith et al. 2015), the Community-based ontology feedback evaluation phase corresponds to the “Detecting the Need for Evolution” phase. In this activity, Zablith’s ontology evolution cycle makes a distinction between bottom up detection approaches which employ automated techniques to analyze data sources (e.g. text, databases) and ontology application usage documents, and top-down approaches where changes are dictated by domain experts. In CMOE+ both types are combined, as the model ontology can be considered as an external data source that contains new terms and relations relevant for ESO evolution, and the log as a usage document of the ESO indicating how the ESO interacts with the modeler.

As community members are typically not co-located at the same physical location, and the aim is to progressively reach a consensus about what is needed by the community, the Delphi approach (Gupta & Clarke 1996) is used. This approach is perfectly suited to capture collective knowledge and experience of experts in a given field, independently of their location, and to reach a final conclusion by consensus. More specifically, consensus is reached by commenting on the feedback in 3 cycles. Three cycles were chosen because studies show that most changes in responses occur in the first 2 rounds (Gupta & Clarke 1996). In every cycle comments are assigned a score, and when all three cycles are accomplished, a decision is taken whether to incorporate feedback or discard it. Only in case the community is not able to reach a consensus, the final decision is made by community members with a high level of trust. A system for assigning trust credits to community members is foreseen. If negotiation upon the feedback progresses slowly, the process may be terminated without accomplishing the predefined number of cycles. In this case highly trusted, authorized community members are responsible to make a decision. Another useful approach is DOGMA-MESS, a collaborative ontology engineering method that contains a relevance scoring mechanism used to create consensus between different experts (de Moor et al. 2006).

2.3.1.4 Ontology evolution

For this phase of CMOE+, only the general guidelines are established in this thesis. More concrete instantiation is a future work.

After the feedback verification is performed, the ontology engineer incorporates the approved feedback into the enterprise ontology following Zablith's ontology evolution cycle. This procedure is executed in four steps. First, the suggested ontology changes are translated into concrete ontology change operations. For instance, the ontology engineer needs to determine the place of a new concept in the ESO hierarchy, and the relations with existing concepts. Second, the ontology changes need to be validated by the ontology engineer at two different levels: 1) domain-based validation and 2) formal properties based evaluation. The domain-based validation by the ontology engineer complements similar analyses executed by the domain experts in the previous phase of the ontology engineering cycle of the CMOE+ framework. The formal properties based evaluation evaluates whether the proposed changes do not violate the imposed integrity constraints of the ESO. Third, the ontology engineer needs to evaluate how the performed ontological changes will impact external artefacts that are dependent on the ontology. Additionally, costs and benefits of performing the changes can be analyzed. Fourth, the ontology engineer needs to record the performed change in order to make it possible to restore a previous version of the ESO when needed and trace back the history of the ontological entities. For all these steps different techniques and tools have been proposed in the ontology evolution literature.

It is worth mentioning that the ontology engineer does not interfere in feedback verification. Their mission is limited solely to incorporating the final results into the ontology in a syntactically correct manner. Once a considerable amount of feedback is incorporated, a new ontology version is proposed. The new ontology incorporates new concept/relationships, updates lacking/incomplete ones, and/or removes irrelevant once, as the domain evolves or new insights are reached by the expanding community.

2.3.2 Conceptual Modeling Cycle

This cycle incorporates three phases: Ontological analyses of the modeling language, Conceptual model creation, and Conceptual model evaluation.

2.3.2.1 Ontological analyses of the modeling language

This is the first phase of the Conceptual Modeling Cycle. During this phase, the ontology engineer performs ontological analysis of the modeling language selected for model creation. The modeling language is analyzed using the same foundational ontology selected in the Ontology Evolution cycle.

The idea of ontological analyses of conceptual modeling languages was introduced in the eighties and has since then frequently been used to evaluate and reengineer conceptual modeling languages. The pioneers in ontological analyses were Wand and Weber as they constructed and applied the Bunge Wand Weber (BWW) ontology (Wand & Weber 1988). Later, BWW was widely used to analyze modeling languages. For example, it was utilized by

Opdahl and Henderson – Seller (Opdahl & Henderson-Sellers 2002) to analyze UML, and by zur Muehlen et al (zur Muehlen et al. 2007) to analyze Simple Rule Markup Language (SRML) and Semantics of Business Vocabulary and Business Rules (SBVR). Prevalently, ontological analyses were used as a tool to identify flaws in the semantics of the modeling language. Santos et al (Santos et al. 2013) identify two main reasons for this based on the literature: (1) conceptual models are created with the aim to represent a part of reality according to a certain conceptualization, (2) foundational ontology classifies domain – independent categories which can be used to articulate those conceptualizations of reality. Hence, conceptual modeling languages need to offer modeling elements reflecting conceptual categories defined in the foundational ontology. In literature, ontological analyses of modeling languages are used for the following purposes:

- To provide a rigorous definition of the construct of a modeling languages in terms of real-world semantics
- To identify inappropriately defined constructs
- To recommend language improvements which reduce lack of expressivity, ambiguity, and vagueness (Almeida & Guizzardi 2013)
- To develop guidelines for the usage of modeling language constructs (Guizzardi & Wagner 2010b)
- To facilitate automated reasoning (Kaneiwa et al. 2007)
- To promote integration and interoperability of different modeling languages analyzed with the same foundational ontology (Harzallah et al. 2012)
- To combine modeling languages to obtain an optimal representation for a given domain (Harzallah et al. 2012)

According to Wand and Weber (Wand & Weber 2002), there needs to be a one-to-one correspondence between concepts of the foundational ontology, and constructs of the conceptual modeling language. If this correspondence cannot be achieved, the modeling language is likely to contain the following semantic issues:

1. *Construct excess*: occurs when a modeling language construct does not correspond to any concept of the foundational ontology
2. *Construct overload*: if particular modeling construct corresponds to several ontological concepts
3. *Construct redundancy*: when several modeling constructs can be represented by a single ontological concept
4. *Construct deficit*: exists when there is an ontological concept which cannot be covered by any of the modeling constructs

In addition to the “conventional” usage of the foundational ontology in analyzing conceptual modeling languages, some foundational ontologies were specifically created to contribute directly to a particular domain instead of analyzing the modeling language used to model this domain. For example, Pedrinaci et al (Pedrinaci et al. 2008) have established a core ontology for enhancing Business Process Analyses domain. Jureta et al (Jureta et al. 2009) created an ontology for Requirements Engineering. Kaniewa et al (Kaneiwa et al. 2007) proposed a foundational (upper) ontology to capture events. And Nardi et al developed a service ontology based on UFO core ontology (Nardi et al. 2013). A weakness of ontological analyses is that it is largely an informal process, and different ontological analyses are not comparable together (Gehlert & Esswein 2007).

As mentioned above, ontological analyses are typically used for analyzing the semantics of the modeling languages, identifying flaws, and proposing alternatives. However, within the CMOE+ framework, ontological analyses of the modeling language are used for a different purpose. CMOE+ does not aim at evaluating the semantics of modeling languages. Selection of the conceptual modeling language is up to the modeler and CMOE+ does not intend to show the flaws of the modeling language. Within this framework, the foundational ontology offers domain-independent semantics which bridges between the ESO and the modeling language, thereby enabling the operation of modeling-language based recommendation services.

Within CMOE +, the task of executing an ontological analyses of the modeling language falls on the shoulders of the ontology engineer. For some modeling languages the analyses are already available in literature can be reused. Some examples are analyses of UML in (Opdahl & Henderson-Sellers 2002) , and of BPMN by Guizzardi and Wagner (Guizzardi & Wagner 2011). Otherwise, the ontology engineer performs this task based on their understanding of the semantics of foundational ontology concepts, and the semantics of constructs of the modeling language. Performing ontological analyses is a difficult task. Here are the main challenges faced (Harzallah et al. 2012):

1. Foundational ontologies are high – level and complex artefacts, which makes them hard to understand
2. There are several well established foundational ontologies, and the result of choosing one over the other cannot be objectively assessed
3. Sometimes constructs of modeling languages do not have a formal and precise definition. Hence, it can be challenging to comprehend the semantics behind it, and its possible usages.

If the ontology engineer was not able to reuse existing ontological analyses, the literature can still be a source of support; there are available frameworks which guide the ontology

engineer during the ontological analyses task. An example is (Gehlert & Esswein 2007) which formalizes ontological analyses as a set of mathematical functions between ontological concepts and modeling constructs, and suggests formal requirements for ontological analyses to comply. Another example is the UEML approach suggested by Harzallah et al (Harzallah et al. 2012) which offers rules and guidelines for the ontology engineer to follow. Every modeling language needs to be analyzed only once. Afterwards, the analyses are reused.

CMOE+ supports any foundational ontology without restriction. While experimenting with CMOE+, we found the Unified Foundational Ontology (UFO) to be the best fit. A detailed description of this ontology is found in (Guizzardi 2005). UFO was chosen because it was already used to analyze many modeling languages and approaches, which increases the chances of reusing existing analyses instead of taking the burden of performing them from scratch. For instance, UFO was used to analyze Business Process Modeling (BPMN) (Guizzardi & Wagner 2011), Unified Modeling Language (UML) (Guizzardi 2005), ARchitecture for integrated Information Systems (ARIS) (Santos et al. 2013), Recourse-Event-Agent (REA) (Gailly et al. 2009), and goal modeling (Guizzardi et al. 2008). Despite the previously mentioned advantages of UFO, it is still complex and hard to understand specially if you are not an experienced ontology engineer. Ontological analyses of modeling languages no doubt has advantages. However, its success depends on the selected core ontology and the correctness of the analyses.

As CMOE+ is used for more modeling languages, the ontological analysis of those different modeling languages will be formalized as OWL ontologies, and stored in the system for reuse. One ontology is created for every modeling language and will be reused for all conceptual models created in this language. Such an ontology contains the mappings between the constructs of the modeling language, and the concepts of the selected foundational ontology.

More practical details regarding this phase are presented in Section 3.3.1.

2.3.2.2 Conceptual model creation

The conceptual model creation phase is responsible for creating conceptual models based on the ESO. It utilizes the recommendation services described in the Ontology storage phase of the Ontology Evolution cycle, and is performed by the modeler. It is important to keep in mind that the modeler may not be acquainted with the concept of “ontology”, and this should not influence the modeler’s ability of benefiting from CMOE+. Hence, the utilization of the ESO needs to be encapsulated in a user-friendly interface which conceals any complications related to ontology usage.

During this phase, the modeler proceeds with creating a conceptual model based on the ESO. With every modeling element placed on the canvas, the modeler receives an ordered list of

ESO concepts. The order in the list is determined based on the weighted average of the scores every ESO concept acquires from the recommendation services. ESO concepts with a higher score are positioned on the top of the list (or otherwise distinguished from the rest). Those ESO concepts are considered suggestions for the selected modeling element. The modeler is free to accept or discard the suggestions. Additionally, the modeler has no obligation to select the suggested concepts. He/she can opt for any other ESO concept with a low(er) relevance score. The advantage of ordering concepts in the suggestions list is that the most potentially relevant concepts come first, thereby saving the time of the modeler from looking through the whole ontology. The suggestions can also be perceived as a hint to the modeler to use a particular ESO concept as a label for the selected modeling element. If the modeler chooses one of the ESO concepts (a suggestion, or a concept with a low relevance score), the selected modeling element is automatically annotated with the corresponding ontological concept. Annotation of modeling elements with ESO concepts is an important part of achieving semantic alignment among conceptual models created within an enterprise as it keeps a semantic link between models and ESO.

When the modeler annotates a modeling element, its label is updated with the name of the selected ESO concept. However, the modeler always has an option to rename - change the label of - the modeling element. Even if the label is changed, the connection with the relevant ESO concept is maintained due to the annotation. If none of the ESO concepts is suitable for a particular modeling element, the modeler assigns a label of his choice and skips the annotation.

A significant amount of work on semantic annotation of conceptual models is rooted into the Semantic Process Management field (Wetzstein et al. 2007; M Hepp et al. 2005) which aims to improve the level of automation in implementation, execution, and monitoring of business processes by enriching process models and other process artefacts with semantic annotation. Even though this discipline is focused solely on process models, it offers some useful ideas which CMOE+ can reuse. According to Thomas et al (Thomas & Fellmann M.A. 2009), semantic modeling offers several advantages:

- It facilitates *distributed modeling*. Annotating processes with concepts from a well-established ontology leads to a unified interpretation of models by different people
- It improves *model management*. Model annotation systematized access and supports search and selection of models. It makes models machine – readable and processible, and allows more complete results for model search queries
- Contributes to *IT – business alignment*. Semantic annotation improves and simplifies coordination between business and IT by using vocabulary which is accepted by both sides, therefore avoiding misunderstanding of requirements and other documents

In the literature, semantic annotations are found to be applied in two ways: through ontological formalization of the modeling language, such as ontological formalization of the BPMN modeling language presented in (Abramowicz et al. 2007a; Chow 2011; Rospocher et al. 2014). The second approach to augmenting models with semantics is by applying the annotation to the model itself, either during model creation, or by annotating existing models (Di Martino et al. 2016; Vazquez, Martínez, et al. 2013). Even though the ideas from literature were interesting, many were applied after the model has been created thereby requiring additional effort. Additionally, ontological concepts are not always prioritized, which forces the modeler to browse through the complete ontology.

Within CMOE+, the annotation is (automatically) applied to the model directly during the modeling process, by the modeler him/herself when they select the appropriate ESO concept. Hence, normally, no additional effort is required after model creation. However, it is still possible to annotate existing models within the enterprise which were created outside the CMOE+ framework. An easy to use interface and a simplified approach to displaying the ontology allows the annotation to be performed by the modeler herself, without the need for the model analyst such as in (Vazquez, Martínez, et al. 2013). The automated suggestions derived by the recommendation services in the Ontology Evolution cycle ease the burden of selecting the appropriate ontological concept for annotation. Within CMOE+, we are only concerned with annotating the content of the model as it suffices the purpose of the annotation.

It is very important to follow the Conceptual model creation phase as it is prescribed in CMOE+; by utilizing the ESO and by adding annotations. The annotation has two main benefits: First, it improves consistency of labels used to label modeling elements created by different modelers. Models within one enterprise are created by modelers with various backgrounds and mindsets. Every person has their own jargon and modeling style. Depending on the ESO for label creation brings unity to all those modeling styles and patterns. Second, due to the annotation, all models are connected through one central hub: the ESO.

While this phase is operating, two feedback files are being created simultaneously, and are used later in the Community-based ontology feedback evaluation phase from the Ontology Evolution cycles. The first file is the Model ontology (MoO) representing the model being created. This file is also utilized by the Rule-based recommendation service (Ontology storage phase). The second file is the log file which deals more with the suggestions derived by the recommendation services. This file includes:

- Suggestions (recommendations) generating for a particular modeling element

- Accepting a recommendation by annotating a modeling element
- Deletion and replacement of model annotation: which can be an indication that this ESO concept is renamed or is outdated

CMOE+ currently supports semantic annotations using OWL. We have recommended using OWL as an ontology representation language in the Ontology Evolution cycle. OWL proved to be very efficient in this phase as well because the reasoning and query services supported by OWL were able to derive useful suggestions. However, this might be due to the nature of the ontologies used within this project. OWL might not be absolute best for everything. Hence, one needs to take the advice of using OWL cautiously.

In order to be able to demonstrate this phase in practice, a CMOE+BPMN tool was implemented to create BPMN models based on ESO (Section 3.3, Figure 8). The tool incorporates an automatic annotation mechanism (Section 3.2.4). Additionally, the tool creates and store both feedback files explained above in this section. A prototype of CMOE+i* tool (Section 4.4) was implemented to allow creation of i* models.

2.3.2.3 Conceptual model evaluation

During the model quality evaluation phase, the quality of a created conceptual model is evaluated against the quality benchmark of the enterprise. This phase is essential because:

1. It ensures that the created conceptual model corresponds to quality standards and can be further utilized within the enterprise
2. It verifies whether the feedback recorded during the model creation task, is useful and potentially worth incorporating into the ESO. If the quality of the created model is poor, the feedback collected from this model will not be very accurate. Hence, it is best to discard this feedback immediately without circulating it through the Community-based feedback verification phase.

The conceptual model evaluation phase can be initiated by the modeler immediately after model creation, or by any other stakeholder on existing models. This phase is accomplished by running various predefined quality checks on the model. Some of those checks are performed automatically, some require human intervention. CMOE+ does not prescribe any particular model quality evaluation framework; quality requirements depend on the type of the created models and their usage. Hence, there is no universal framework, and every enterprise has the freedom to use their own quality requirements.

Model quality is an important part of conceptual modeling research, and many frameworks and guidelines have been published. Every framework has its own advantages, motivation, and focus. Some frameworks are generic, and can be used with any type of conceptual model (Maes & Poels 2007; Nelson et al. 2011; Shanks & Darke 1997). Other quality

frameworks are associated with a particular modeling domain, but independent of a particular notation such as (Ayad 2012; Becker et al. 2000; Sánchez-González et al. 2013) for the Business Process Modeling domain, and (Levitin & Redman 1995) for Data Modeling. The third set of quality frameworks is dedicated to a specific modeling notation. For example, Kesh proposes a quality metrics for evaluating Entity Relationship ER models (Kesh 1995), while Rolon et al evaluate Business Process Modeling Notation BPMN (Rolón et al. 2006). The frameworks mentioned above focus particularly on the final product of modeling (the conceptual model itself). However, researchers such as Moody et al (Moody et al. 1998) argue that the process of model creation also has a considerable impact on the end result. Hence, one can find frameworks on monitoring the quality of the task of modeling. One example is offered by Wand and Weber where they describe a process of conceptual modeling based on the Bunge's ontology (Wand & Weber 1990). Another framework for evaluating the quality of the modeling process is described in (Bommel & Hoppenbrouwers 2007).

It is important to note that CMOE+ framework imposes a degree of quality control on the process of modeling, as the modeler is guided by ESO during model creation. However, we do not prescribe any particular frameworks for evaluating the process of modeling. Regarding the evaluation of the end product, the conceptual model itself, one possibility is adoption of a well-known scheme for classifying quality dimensions, the Lindland et al framework (Lindland et al. 1994) which makes a distinction between syntactic, semantic and pragmatic quality dimensions. *Syntactic quality* implies correspondence between the model and the modeling language. *Semantic quality* measures how compliant the model is to the domain. And *pragmatic quality* is the correspondence between the model and the user interpretation of it.

Coming back to the two points emphasizing the importance of this phase (in the beginning of this section), the first importance can be satisfied by focusing on measures that fall within the syntactic and pragmatic dimension because for the model to be used within the enterprise it needs to be correct and understandable. The second point of significance can be achieved mostly by focusing on measures that fall within the semantic quality dimension. More specifically, the developed model has to correctly reflect the part of the enterprise it was designed for. The fact that a model possesses high semantic quality implies that it faithfully represents its domain. Therefore, the feedback gathered from such a model is useful for ontology improvement. On the other hand, if a model scored low on semantic quality, it means that it does not completely satisfy its purpose and reflect its domain. Consequently, the feedback gathered from this model will not have a valuable contribution to ESO enrichment. Pragmatic quality also has a role to play in this aspect; if the model is hard to understand, it consequently will be challenging to evaluate the feedback.

This phase has a rather limited scope within this thesis, as it is completely up to the enterprise which employs CMOE+ to determine which quality evaluation framework they desire to utilize.

2.4 Correspondence Between CMOE+ Requirements and Phases

Table 2 below iterates on the requirements presented in Section 2.2, summarizes the importance of every requirement and its added value to the whole framework. The third column explains design decisions we took to address every requirement while developing the CMOE+ framework. The last column shows which phase(s) of CMOE+ are responsible for each requirement.

Table 2: Requirements for the proposed framework

<i>Requirement ID*</i>	Importance	Design Decision	Responsible phase(s) of CMOE+	Progress
REQ1	This requirement is important as it ensures covering all aspects of an enterprise, including various modeling languages and modeling approaches such as data modeling, goal, and process modeling	The framework is kept very flexible and example instantiations are provided. The recommendation services which derive ESO suggestions include mechanisms operating independently of the modeling language used. Rule – based recommendation service allows configuration of new rules at any stage of CMOE+	All phases of the Conceptual Modeling Cycle	Addressed within this PhD
REQ2	Saving time and effort and facilitating ontology lookup	Relevance on ontological concept is calculated using a set of configurable algorithms. ESO concepts are presented to the modeler in an ordered list where concepts with higher	<ul style="list-style-type: none"> • Ontology Storage • Conceptual Model Creation 	Addressed within this PhD

		scores are on the top		
REQ3	Ensuring that organizational quality standards for conceptual modeling are maintained	Incorporation of a model quality evaluation framework, and prohibiting low quality models to proceed through further phases of CMOE+	Conceptual Model Evaluation	Quality frameworks are available in the literature, and this phase will not be elaborated any further
REQ4	Ability to proactively capture any changes to the enterprise which are presented in the conceptual models being created. For any change to be captured by CMOE+, it first needs to feature in one of the conceptual models created using CMOE+	Feedback from conceptual models is stored automatically while models are being developed. This feedback later will be used to search for relevant updates	Conceptual Model Creation	Future work
REQ5	Ensuring that the selected feedback indeed reflects relevant changes within the enterprise, and not only presenting some localized, or temporary changes	CMOE+ incorporates a community guidance framework which dictates how the feedback is to be reviewed	Community – Based Ontology Feedback Evaluation	Future work
REQ6	The purpose of establishing the ESO at the first place, is for it to become a reference during model creation. Hence, it needs to faithfully represent the enterprise, and reflect the changes occurring within that enterprise	The framework offers support to the ontology engineer during feedback incorporation and management of various versions of the ontology	Ontology Evolution	Future work
REQ7	This requirement	Semantic links are	Conceptual Model	Addressed

allows querying the models, and handling issues related to ontology versioning	created by means of semantic annotation. Each element of a conceptual model is annotated with ESO concept selected by the modeler	Creation	within this PhD
--	---	----------	-----------------

*Requirement IDs correspond to the IDs used in Section 2.2

2.5 Conclusion

This chapter presents an overview on how the literature tackles the problem of semantic alignment of conceptual models. It puts forward the requirements for CMOE+ and describes the different phases of the framework. The summary of the phases of CMOE+ and “how to proceed” is presented in Table 3 below.

Table 3: Summary of different phases of CMOE+

CMOE+ Phase	WHAT is to be accomplished	HOW to proceed
<u>Phase 1:</u> <i>Ontology setup</i>	<ul style="list-style-type: none"> • Selecting the ESO and the foundational ontology to be used throughout the method • Grounding ESO in the foundational ontology 	<ul style="list-style-type: none"> • In the ideal situation, the ESO is already available within the enterprise. If this is not the case, the ontology engineer needs to look into available business artefacts such as glossaries, vocabulary, etc., or if there is an existing domain ontology which can be reused. If nothing of the above is present, ESO is constructed from scratch following existing ontology engineering practices • While selecting the foundational ontology, there are two criteria to consider: first, its usefulness for grounding ESO (or domain) ontologies. Second, the availability of ontological analyses of the modeling languages used within the enterprise with this foundational ontology • The grounding is performed by an ontology engineer. Existing

		research can be used to verify the correctness of the mappings (optional)
<i><u>Phase 2:</u> Ontological analyses of the modeling language</i>	Performing ontological analyses of the modeling language to be used for model creation with a foundation ontology selected in Phase 1	<ul style="list-style-type: none"> • Look for literature including ontological analyses of the selected modeling language • If literature is not available, the ontology engineer can perform the analyses based on his understanding of the semantics of the modeling language constructs
<i><u>Phase 3:</u> Ontology storage</i>	<ul style="list-style-type: none"> • Selecting the ontology representation language • Choosing ontology representation medium • Selecting the optimal combination of the recommendation services to identify the most relevant ESO concepts to be used for annotating a particular modeling element 	<ul style="list-style-type: none"> • Carefully examine all the relevant factors before opting for a particular ontology representation or storage medium • Review the algorithms for the recommendation services and make the necessary adjustment according to the chosen foundational ontology and conceptual modeling language, such as configuring a dictionary or a customized database for retrieving synonyms, and adding matching rules to be used by rules – based recommendation service
<i><u>Phase 4:</u> Conceptual model creation</i>	Creating conceptual models based on the ESO	<ul style="list-style-type: none"> • Carefully examine the ESO recommendations list to choose the appropriate ESO concept for every modeling element • Annotate modeling elements with ESO concepts
<i><u>Phase 5:</u> Conceptual model evaluation</i>	Measuring quality of conceptual models against predefined quality standards	<ul style="list-style-type: none"> • Run predefined quality checks against the created model • Update the model in case of reported insufficient quality
<i><u>Phase 6:</u> Community – based ontology feedback evaluation</i>	Evaluating the feedback gathered during model creation. This feedback has a potential to be used to update the ESO if proved useful	The ontology engineer extracts the feedback from the log file, and makes it available to the community members. The community negotiates upon the feedback following a predefined protocol

<i>Phase 7: Ontology evolution</i>	Updating the ESO using the feedback which deemed to be useful in the previous phase into the ontology, and creating a new ontology version	<ul style="list-style-type: none"> ○ Execute all the necessary steps to correctly reflect the obtained feedback into the ESO. Those steps can be adding / deleting a concept, adjusting a definition, adding a relationship, etc. ○ Manage the versioning of ESO and ensure compatibility of previously created models
--	--	--

This chapter puts forward requirements for the framework responsible for establishing the symbiosis between conceptual modeling and ontology engineering. Not all phases of this framework were explored within this PhD. However, when CMOE+ is fully implemented, it is expected to deliver the following features:

1. It is suited for different modeling languages, thereby allows creation of different types of (semantically annotated) models. Hence CMOE+ promotes semantic alignment of different model types covering all organizational needs, not only a particular subset.
2. The framework ensures proper maintenance and enrichment of the ontology according to the emerging needs of the enterprise.
3. CMOE+ prescribes guidelines on how to evaluate the candidate feedback before ontology enrichment. The feedback first needs to be processed by ontology engineer who extracts candidate feedback, then this candidate feedback is thoroughly assessed by enterprise community members. The approved feedback is used for ontology amelioration, and the new version of ESO is evaluated for quality.
4. Recommendation services incorporated within the framework assist the modeler during the modeling process in selecting the appropriate ESO concept for model annotation. Recommended ESO concepts appear on the top of the ESO concept list, so that the modeler browses them first. If the appropriate concept is not found, the modeler proceeds through the whole ontology. The ESO can be very extensive and selecting a subset of the most relevant concepts can significantly save time and effort.

Within the context of this work the focus was shifted towards the Conceptual Modeling cycle (Figure 3). This research contributes by (1) Offering precise guidelines and description of phases related to the conceptual modeling process, and (2) Implementing the previously mentioned phases in a tool for BPMN (Chapter 3) and i* modeling languages (Chapter 4),

and offering an evaluation (Chapter 3). Additionally, the following contributions are realized within this chapter:

- Developing a comprehensive set of recommendation services covering different aspects including label similarity, the meta model of the modeling language, and more. Recommendation services are essential in supporting the modeler and facilitating access to the ESO.
- Offering a mechanism for an easy to use semantic annotation. Linking modeling elements to ESO concepts is an important aspect of the successful operation of CMOE+. This thesis proposes a mechanism using which the modeler can perform annotation without the need to be completely aware of what the semantic annotation represents. Additionally, it does not require knowledge on ontologies.
- Utilizing ontological analyses in practice. Ontological analyses are widely used in theory, and this work allows practitioners to benefit from those analyses by setting the guidelines and offering demonstration.
- Experimenting with different ontology storage and utilization mechanisms.

As was mentioned throughout this chapter, CMOE+ relies on the literature, and aims to utilize what is already available. The framework reuses available foundational ontologies and established ontological analyses. The target audience for CMOE+ are real life enterprises, and the purpose is to make it as easy to use as possible within the settings of the enterprise. An enterprise might not possess a qualified ontology engineer as its staff member. Hence, it is important to reuse as much as possible already established material from the literature. Another aspect that was reused is the idea of utilizing the semantic annotation as means to link the conceptual models to the ESO. Finally, several independent matching algorithms were reused within the recommendation services. However, in addition to those reused algorithms, the recommendation services incorporate some novice algorithms which did not appear in the literature previously (to the best of our knowledge).

CMOE+ offers detailed description and demonstration regarding the phases related to the Conceptual Modeling activity. Phases concerned with the Ontology Evolution remain empty boxes which will be addressed in detail in another project.

3

Recommendation-Based Process Modeling

3.1 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 modeling 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; J Becker et al. 2009).

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 modeling process by suggesting a limited set of enterprise concepts and relationships. However, it also constrains the freedom of the modeler, 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; J Becker et al. 2009), annotate the model elements (Thomas et al. 2009; Born et al. 2007), 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 modeling elements or language are imposed. The drawbacks are the lack of guidance during modeling 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 modelers 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 Conceptual Modeling and an Ontology Evolution 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 modeler 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 modeling suggestions. CMOE+ thus establishes a symbiotic relationship between conceptual modeling, 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 modeling 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 modeling language, but instead is designed to support existing, well-known modeling 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 modeling 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 modeling languages. In this article, we present one such concrete (partial) instantiation, CMOE+BPMN, which provides recommendation-based modeling support for business process modeling (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.

3.2 Recommendation-Based Conceptual Modeling and Ontology Evolution (CMOE+) Framework

The CMOE+ framework was conceived through the Design Science Research Methodology (DSRM) (Hevner et al. 2004), 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 10. The Java implementation of the CMOE+ framework is publicly available (Gailly 2016). It consists of two cycles, the Conceptual Modeling (CM) and Ontology Engineering (OE) cycle, and establishes a symbiotic relationship between these. This paper describes the development and evaluation of the ontology-assisted modeling 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 modeling languages, the ontology storage, the recommendation services, and the model creation phases of the CMOE+.

3.2.1 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 modeling 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 (Ruy et al. 2015; Blomqvist 2005). Other useful tools for ontology engineers are ONTOCLEAN (Guarino & Welty 2002b) and OntoUML (Guizzardi

et al. 2015), which can be used to evaluate the grounding of ontology concepts in the core ontology.

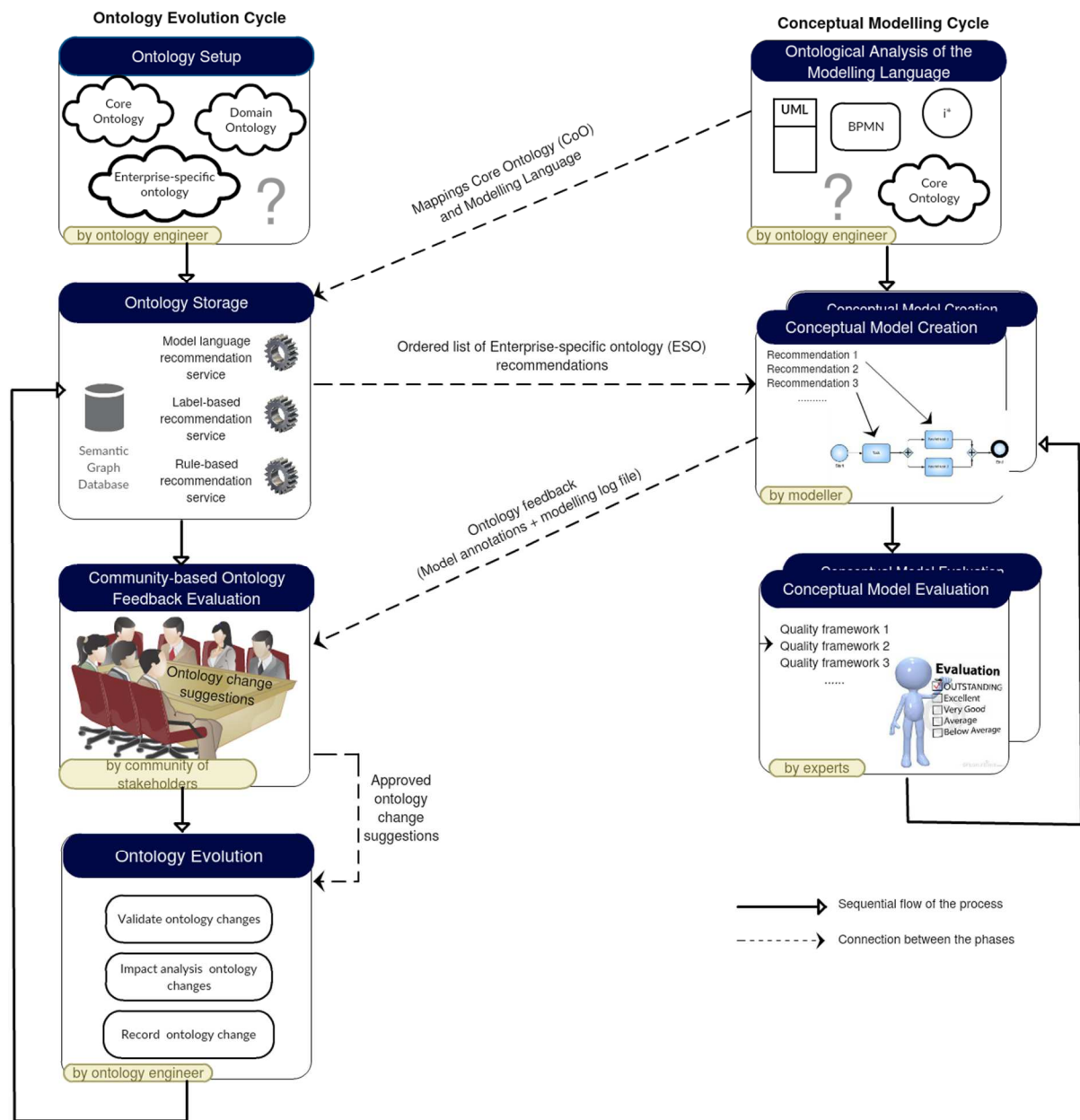


Figure 10: Recommendation-based Conceptual Modeling and Ontology Evolution (CMOE+) Framework

3.2.2 Ontological Analysis of the Conceptual Modeling Language

The first phase of the conceptual modeling cycle is another initialization phase, in which an ontological analysis is performed for the target conceptual modeling language(s) used in the enterprise. Different authors have proposed methodologies and frameworks to achieve this (Guizzardi 2013; Harzallah et al. 2012; Evermann & Wand 2005b). The purpose of these methodologies and frameworks is (1) to provide a rigorous definition of the construct of a modeling 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 modeling language to the core ontology selected in the ontology setup phase. These connections can later on be exploited in the conceptual modeling recommendation service (see section 3.4).

Over the years, different conceptual modeling 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 modeling 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).

3.2.3 Ontology Storage and Recommendation Services

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)⁷ 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 database⁸, 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 modeling 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.

The Stardog Database consists of different interconnected OWL ontology files. Panel A of Figure 11 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.

⁷ <https://www.w3.org/TR/owl2-overview/>

⁸ <http://www.stardog.com>

- The Modeling Language Ontology (MLO) file is a formalization in OWL of the meta-model of the used conceptual modeling 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 modeling 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 11 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 modeling language construct that the modeler 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).

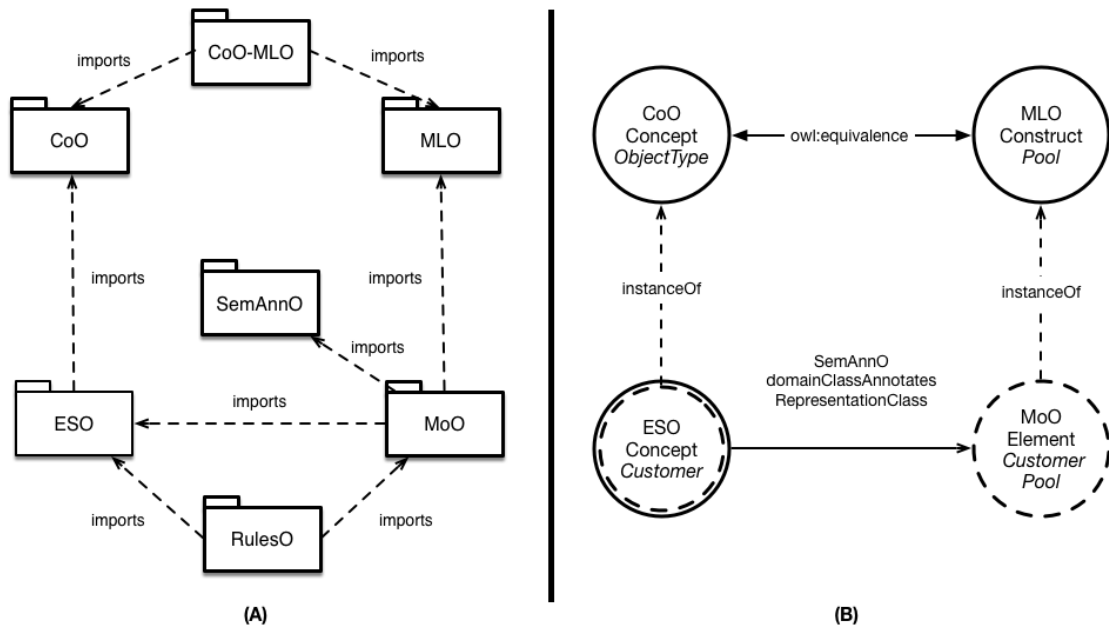


Figure 11: Ontologies within CMOE+ framework

Recommendation Services

Based on the above-mentioned stored ontologies, the recommendation services determine what ESO concepts are suggested to the modeler. For each ESO concept, each recommendation service calculates a recommendation score between 0 and 1, with respect to a modeling element added by the modeler. 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 modelers 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 modeling language: given a modeling 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 modeling 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  $\leftarrow$  CoO  $\cup$  MLO  $\cup$  CoO-MLO  $\cup$  ESO;
3   ontology  $\leftarrow$  Reasoner.reason(ontology);
4   query1  $\leftarrow$  SELECT ?x WHERE {?x rdf:type :MLconstruct };
5   ESOcandidates  $\leftarrow$  Reasoner.query(ontology, query1);
6   foreach element  $\in$  ESOcandidates do
7     query2  $\leftarrow$  SELECT ?y WHERE {?y rdfs:subClassOf :element};
8     subClasses  $\leftarrow$  Reasoner.query(ontology, query2);
9     ESOcandidates  $\leftarrow$  ESOcandidates  $\cup$  subClasses;
10  end
11  if ESOConcept  $\in$  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. ESO candidates) 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 modeler. Listing 2 presents the pseudo code. In Line 3, the string matching score is calculated using Jaro-Winkler the label that is entered by the modeler 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  $\leftarrow$  getLabelConcept(ESOconcept);
3   score  $\leftarrow$  getStringMatchScore(conceptLabel, enteredLabel);
4   synonyms  $\leftarrow$  getWordNetSynonyms(conceptLabel);
5   foreach synonym  $\in$  synonyms do
6     score  $\leftarrow$ 
7       max(score, getStringMatchScore(synonym, enteredLabel))
8   end
9   return score
10 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 modeling element added by the modeler. Listing 3 presents the pseudo code. The algorithm starts with creating a new modeling element (see Line 2) which corresponds to the model element that is currently selected by the modeler 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  $\leftarrow$  createElement(MLconstruct);
3   MoO'  $\leftarrow$  add(MoO, newElement);
4   ontology  $\leftarrow$ 
       CoO  $\cup$  MLO  $\cup$  ESO  $\cup$  MoO'  $\cup$  CoO-MLO  $\cup$  RulesO  $\cup$  SemAnnO;
5   ontology  $\leftarrow$  Reasoner.reason(ontology);
6   query  $\leftarrow$  SELECT ?x WHERE
       { :hasSemAnn rdfs:domain :newElement rdfs:range ?x };
7   result  $\leftarrow$  Reasoner.query(ontology, query);
8   if ESOconcept  $\in$  result then
9     | return 1;
10  else
11    | return 0;
12  end
13 end

```

Listing 3: Pseudo-code rule-based recommendation service

3.2.4 The Conceptual Model Creation Phase

In the *Conceptual model creation phase* (CM cycle), the modeler is presented with an ordered list of ESO recommendations, based on the selected modeling 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 modeler. The modeler 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 modeling 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 modeling 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 tool⁹. 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.

⁹ <http://www.promtools.org>, last accessed 5 August 2016

3.3 Recommendation-Based Business Process Modeling (CMOE+BPMN)

To demonstrate that the CMOE+ framework is a feasible, adequate and efficient solution for the presented problem, it was instantiated for process modeling by means of BPMN. Consequently, we will now move on to describe the CMOE+ recommendation-based business process modeling 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 GitHub¹⁰ and is shown in Figure 12. By means of the Eclipse plug-in extension point mechanism, the CMOE+BPMN plug-in extends the Eclipse BPMN2 modeler¹¹ with two views and a preference page. BPMN2 Modeler is a graphical modeling 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.

¹⁰ <https://github.com/fgailly/CMOEplusBPMN>, last accessed 5 August 2016

¹¹ <http://www.eclipse.org/bpmn2-modeler/>, last accessed 5 August 2016

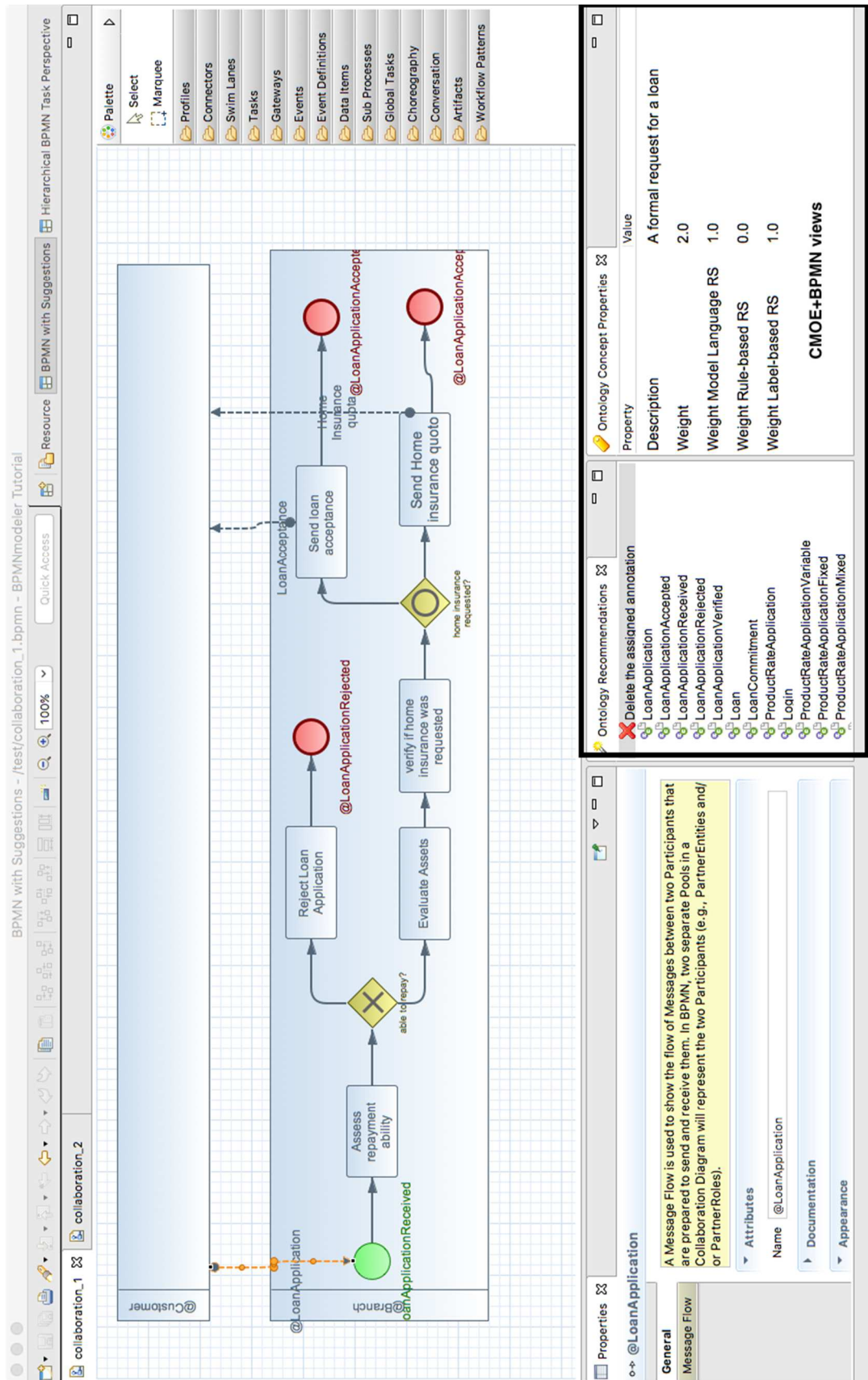


Figure 12: BPMN tool

3.3.1 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 context of process modeling for this instantiation of CMOE+. A short description of UFO can be found in Appendix C; for a full explanation, we refer to Guizzardi et al. (2015). The OWL formalization of UFO is available online¹².
- 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 D, 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 online¹³.
- The used BPMN ontology (i.e. MLO) is an OWL translation of the meta-model shown in Figure 13, and is based on the original OMG BPMN standard (OMG 2006b). In this paper, we extend OMG meta-model based on the observation that different authors advise BPMN modelers 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 online¹⁴.
- The mappings between UFO and BPMN (i.e. CoO-MLO) are based on the ontological analysis provided by (Guizzardi & Wagner 2011). Table 4 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.

¹² <http://www.mis.ugent.be/ontologies/ufo.owl>, last accessed 5 August 2016

¹³ <http://www.mis.ugent.be/ontologies/bank.owl>, last accessed 5 August 2016

¹⁴ <http://www.mis.ugent.be/ontologies/bpmn.owl>, last accessed 5 August 2016

contracts, invoices), and Base types (e.g. database, technical documentation of software). The OWL formalization of the mappings is available online¹⁵.

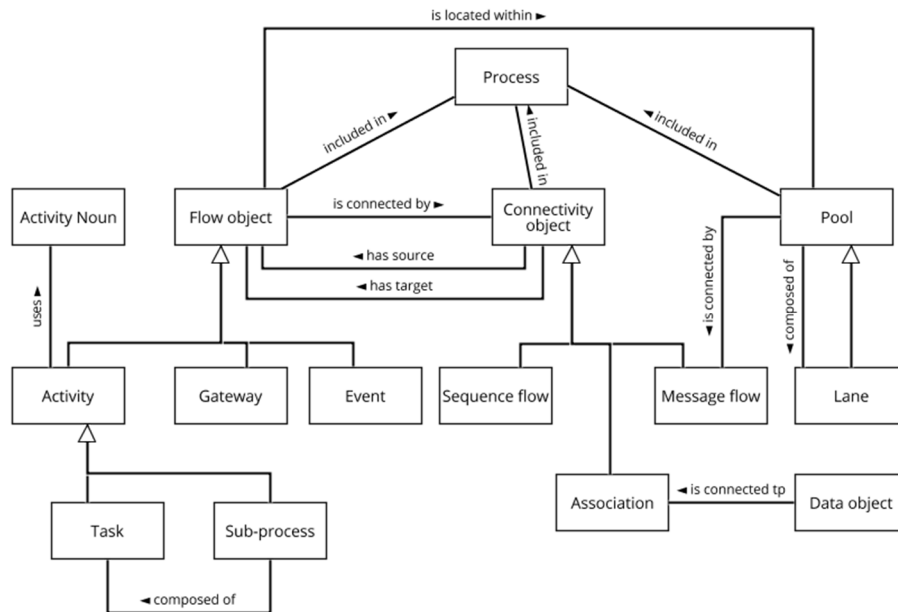


Figure 13: BPMN meta-model

Table 4: Correspondence between BPMN and UFO

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

3.3.2 Recommendation Services

The recommendation services are used by the BPMN editor to arrange the ESO concepts in the ontology property view (see Figure 14), which is implemented following the Model-View-Controller pattern. The controller of the ontology recommendation view updates the associated view every time the modeler selects a model element on the canvas. The 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 modeler selects an ontology recommendation.

¹⁵ http://www.mis.ugent.be/ontologies/bpmn_ufo.owl, last accessed 5 August 2016

CMOE+BPMN uses the OWL API¹⁶ 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 Parser¹⁷ is applied to tokenize the labels.

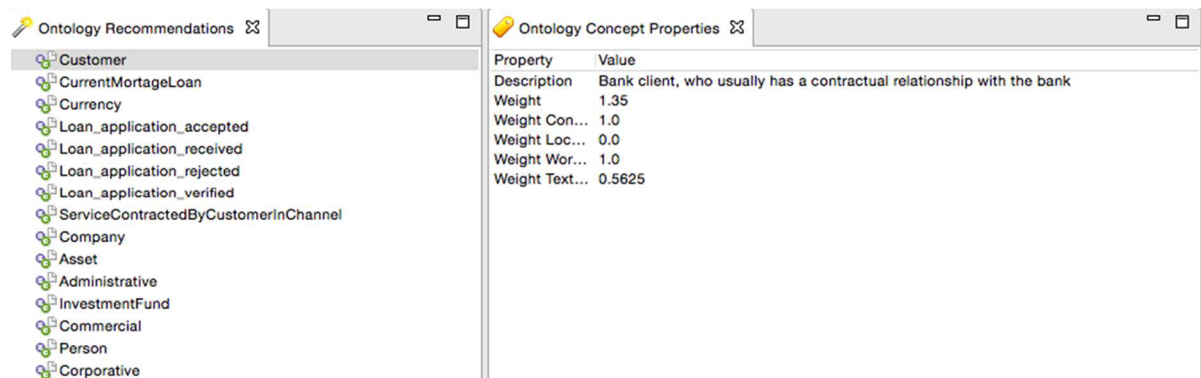


Figure 14: Ontology Recommendation view (Left) and Ontology Concept Properties view (Right)

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 5; a full specification can be found online¹⁸. In future research, we plan to investigate in more detail which kind of rules may be useful to add to this recommendation service.

3.4 Evaluation of CMOE+BPMN

CMOE+BPMN aims to promote label consistency and facilitate model annotations, while ideally avoiding significant overhead in modeling time and perceived effort. Annotating modeling 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).

¹⁶ <http://owlapi.sourceforge.net>, last accessed 5 August 2016

¹⁷ <http://nlp.stanford.edu/software/lex-parser.shtml>, last accessed 5 August 2016

¹⁸ http://www.mis.ugent.be/ontologies/cme_bpmn_rules.owl, last accessed 5 August 2016

Table 5: SWRL rules used by the rule-based recommendation service

BPMN:Pool(?x) ^ BPMN:Pool(?y) ^ SemAnn(?x,o) ^ UFO:mediates(?r,o) ^ UFO:mediates (?r,p) ^ \rightarrow SemAnn(?y,p)
<p>This rule indicates that when the modeler 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.</p>
BPMN:Pool(?x) ^ BPMN:Lane(?y) ^SemAnn(?x,o) ^ UFO:mediates(?r,o) ^ UFO:mediates (?r,p) ^BPMN:hasLane(?x,y) \rightarrow SemAnn(?y,p).
<p>This rule indicates that when the modeler 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.</p>
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) \rightarrow SemAnn(?x,r).
<p>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.</p>

3.4.1 Experimental Design

Using an identical case description (see Appendix E), modelers were asked to create a BPMN model. Three different treatments were applied: treatment 1 assists modelers with CMOE+BPMN as described in section 4.3; treatment 2 provides modelers an alphabetically ordered list of ESO concepts, without relevance ordering, so that the modeler needs to find relevant ESO concepts him/herself; treatment 3, as a baseline, does not provide any modeling support (i.e. regular BPMN modeling). Where relevant (treatment 1 and 2), the modeler was asked to annotate the modeling element with ESO concepts. The BPMN modeling 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.

3.4.2 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 F and G), collect the created models and record the time spent on each task.

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 modeling experts (Appendix H).

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 modeling 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 G). 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).

3.4.3 Experimental Results

Before analyzing 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 models¹⁹ and reflect a real-life setting in which syntactically incorrect models are improved before acceptance or discarded.

For the retained models, we analyzed 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 continuous 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 6 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 usage, relaxing their focus on the modeling and model completeness. These possible influences should be eliminated in follow-up experiments.

The results for **Interoperability** are shown in Table 6 (number of annotations) and Tables 7 and 8 (naming variation). Considering average and median percentages of annotated modeling elements per treatment (Table 6), roughly 70% of BPMN elements were annotated

¹⁹ 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.

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, modelers annotate a large portion of their modeling elements, thus increasing model interoperability. Furthermore, customized recommendations, as provided by CMOE+BPMN, increase the number of fully annotated models.

Table 6: Results of semantic quality evaluation model annotation

		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 Annotations	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 7 presents the results of naming distribution across models for elements referring to a *customer* (i.e. a single BPMN pool), whereas Table 8 shows the results for three different modeling 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 modeling element. We can observe that for “customer” (Table 7) and “loan application” (Table 8), 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), modelers 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 7: Naming for BPMN elements with underlying meaning “customer”. Columns are modeler-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning “customer”

	<i>Customer</i> (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 8: Naming for BPMN elements with underlying meaning “loan application”. Columns are model-entered labels; rows are treatments; cells denote number of uses of the label / total number of occurrences of BPMN constructs with underlying meaning “loan application”

	<i>Loan application</i> (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 9 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 modeling elements (rather than freely writing a label).

Table 9: 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 use** (IU) are summarized in Table 10, 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 modeling 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 modeling task. Finally, results for **Intention of use** (IU) (see Table 10) do not imply any statistical significant difference.

Table 10: 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 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 modeler 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 modeling 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 modeling does not require extra time, and does not impact on the modeler’s acceptance of the modeling 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 modeling. 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 favor of CMOE+BPMN.

3.5 Conclusions and Future Work

This article introduces the recommendation-based conceptual modeling 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 modeling 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 modeling and ontology evolution within an enterprise.

The framework is instantiated for the BPMN modeling language in a recommendation-based process modeling method (CMOE+BPMN). This instantiation focuses on the modeling 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 modeling language, developing recommendation-based services, and extracting feedback. It was implemented as a plug-in that extends the Eclipse BPMN2 modeler, 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 modeling indeed results in more consistent labelling of modeling 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 modeling 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 modeling method to be more advantageous than vocabulary-assisted modeling. 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 modeling feedback, which has already been gathered and (manually) verified to be useful, in a more formal framework, through a community-based ontology evolution approach.

4

Aligning I* Models and BPMN Models Using the CMOE+ Framework

After demonstrating the performance of CMOE+ framework After instantiating and evaluating the CMOE+ framework for process models in BPMN notation (Chapter 3), the next step is to demonstrate how CMOE+ can contribute to the alignment of models created in different modeling languages. More specifically, this chapter explains how CMOE+ can be used to align goal models in i* with process models in BPMN. This chapter commences by presenting what is available in the literature on goal and process model alignment. Next, the development of the CMOE+i* framework and tool is described. In Section 4.3, our approach on the alignment between i* and BPMN models is presented, along with the extensions added to CMOE+BPMN (Chapter 3). This elaboration is followed by a demonstrative example. Finally, the structure of the CMOE+ tool is explained and visualized.

4.1 Aligning Goal and Process Models

The Business Process Management discipline describes methods, techniques and tools to discover, analyze, redesign, execute and monitor business processes. Business processes and their representation (i.e. business process models) play a crucial role in the different BPM life cycle activities or phases. Additionally, it is generally recognized (Rosemann et al. 2015) that the goals of a company influence the decisions that are made within the different phases of the BPM life cycle. For instance, during the process identification phase a process architecture is developed, which is afterwards used to prioritize the different business processes. This prioritization of business processes is of course very much influenced by the goals of the company. During the process analysis phase, process models can be analyzed by

looking at process performance dimensions: time, cost, quality and flexibility. Again, the importance of every dimension will be determined by the goals of the company.

This chapter aims to investigate the semantic alignment between goal and process models. We consider two models (a goal and a process model) to be semantically aligned when overlapping elements of those models are consistent on the model and the meta-model levels. On the model level, model elements can be annotated with ESO concepts. Overlapping model elements are annotated with the same ESO concept, which ensures consistency of model elements. On the meta-model level, corresponding modeling elements are represented by equivalent modeling constructs in different modeling languages. Construct equivalency is determined by the ontological analyses of the modeling language using the core ontology (Section 2.3.2.1) which acts as an intermediate between both modeling languages (goal and process modeling).

In the course of the last 15 years, different methods have been proposed which focus on aligning goal models and business process models (Nagel et al. 2013; Cardoso et al. 2011; Koliadis et al. 2006; Santos et al. 2010; Lapouchnian et al. 2007; Guizzardi & Reis 2015; Decreus & Poels 2011). There are three main directions pursued by researchers regarding the synergy between goal and process models. The first direction focuses on aligning and ensuring consistency among goal and process models already existing within an enterprise (Nagel et al. 2013; Cardoso et al. 2011; Guizzardi & Reis 2015). In the same regard, researchers aim to maintain the synergy between goal and process models by developing an approach which allows transmitting changes occurring in one model to the other (Koliadis et al. 2006), or by using goal models to discover and reflect organizational changes to process models (Santos et al. 2010). The second direction is deriving one model from the other. For instance, Decreus and Poels (Decreus & Poels 2011) propose a method to automatically derive BPMN models from requirements models, and Lapouchnian et al (Lapouchnian et al. 2007) allow semi – automatic derivation of process models from goal models. Some research works are bi-directional and prescribe rules for deriving models in both direction (deriving process models from goal models and vice versa). An example of this is provided in (Koliadis et al. 2006). Within the third direction, researchers combine goal and process models to form a more comprehensive modeling language capable of addressing tactics, intentionality and operations of an enterprise (Sousa et al. 2014).

As follows from the above, the alignment between goal and process models is very important for organizations, and triggered many research efforts. Hence, goal and process models were selected among other types of conceptual models to evaluate CMOE+'s capability in improving alignment among models in different modeling languages. BPMN (OMG 2011) was selected to formalize process models, as this notation is popular among business analysts (Santos et al. 2010). I* goal modeling notation (Yu et al. 2011) was selected to portray the goal model. Despite i*'s limited usage in practice, it stimulated an enormous amount of research (Moody et al. 2010), and proved to be a very useful and

effective techniques for modeling requirements within research projects held in cooperation with enterprises (Yu et al. 2011).

In CMOE+, the overarching goal is to achieve alignment between the various conceptual models that are created within the enterprise. Model alignment is supported by the CMOE+ framework by means of two mechanisms. First, alignment is partly realized via the ESO, which contains concepts and relations that can be (re)used during the development of different conceptual models, whichever their type or conceptual modeling language used. The impact of this alignment mechanism largely depends on the quality of the ESO (completeness, level of detail, accuracy in reflecting the business and enterprise domain, up-to-dateness) and the consistency and traceability of ESO versions as the ESO evolves over time. In this chapter, we focus on the second alignment mechanism, which exploits the versatility of the rule-based recommendation service, capable of incorporating various sources of knowledge into the recommendation rules. Concretely, we show how existing conceptual models, which were previously created using CMOE+ in the same enterprise context and are stored in the model repository, form the basis for recommendation rules promoting model alignment.

CMOE+ is a generic framework that is capable of operating with a variety of modeling languages. Throughout this chapter, CMOE+ is instantiated with the purpose of establishing alignment between process and goal models by incorporating existing goal and process model alignment techniques found in literature in CMOE+BPMN. BPMN (OMG 2011) was selected for process modeling, and the i* modeling language (Yu et al. 2011) for representing goal models. The proposed alignment method has three main benefit: (1) It is based on existing literature, thereby facilitating reuse of existing work published in the domain. (2) It is flexible and able to incorporate additional rules. CMOE+ does not operate based on a restricted set of predefined rules hardwired into the framework, but can easily accommodate new rules, taken from literature or ad hoc created. Hence, every enterprise can adapt CMOE+ based on its needs and modeling languages used. (3) CMOE+ ensures alignment during model creation, in this chapter, during BPMN model creation. Hence, no extra effort is required to align process models after they are created. Several recommendation services operating within CMOE+ are scanning the ontology (ESO) for the most relevant concepts to be suggested to the modeler during the modeling process. In CMOE+BPMN, one particular recommendation service is accessing a model repository containing previously developed i* models. Those models are queried for the most relevant concepts for the process model being created. Those recommendations guide the modeler through the modeling process.

4.2 CMOE+ I* Tool

Before proceeding with the alignment, it is important to develop the CMOE+i* framework and tool to be able to produce semantically annotated i* models. To develop the CMOE+i* framework, the same steps were followed as in Chapter 3 for CMOE+BPMN.

The same foundational ontology as for CMOE+BPMN, UFO, is used for CMOE+i*. In a real life enterprise, the same ESO is utilized for BPMN and i*, and any other modeling language. The ESO set up phase is required to be performed only once within the enterprise. Hence, this phase does not require any additional effort. However, within this PhD project it was beneficial to experiment with an ontology in a different domain, to show genericity of the framework. Therefore, an existing ontology in a medical emergency domain was selected and slightly adapted to serve as ESO.

The first step requiring some effort is performing an ontological analysis of the i* modeling language using UFO (the foundational ontology recommended in combination with CMOE+). I* is frequently used by researchers, and there existing ontological analyses are available in literature. In this dissertation, the ontological analysis performed by Franch et al (Franch et al. 2011) is used. Table 11 below presents mappings between i* and UFO.

Table 11: Ontological analyses of i*

I* construct	UFO-U concept	I* construct	UFO-U concept
Agent	Physical agent type	Contribution link	Formal relationship type
Role	Role type	Decomposition link	Formal relationship type
Position	Role type	Means ends link	Formal relationship type
Task	Quality universal Relator universal	Goal	Relator universal
Resource	Object type		

The next step is setting up the recommendation services which are scanning the ESO to search for the most relevant ESO concepts for every modeling element created on the modeling canvas. As was explained in Section 2.3.1.2, there are three types of recommendation services:

1. *Label-based recommendation service* targets the label of the modeling element, and matches it to the ESO concept label. This recommendation service includes string and synonym matches. This service remains the same independently of the modeling language used as it only takes into account the string of the label independently of the modeling construct represented by this label. Hence, the same Label-based algorithms can be re-used with both BPMN and i*, and any other modeling language.

2. *Model language-based recommendation service* targets the constructs of the modeling language. This recommendation service is differently instantiated for every modeling language based on the ontological analyses of a particular modeling language. The model language-based recommendation service for i* is based on Table 11 above.
3. *Rule-based recommendation service* incorporates customized rules which are either based on the modeling language used, the location of the modeling element within the whole model, or on generic rules of the enterprise. Most of the rules are modeling language-specific, and therefore need to be created separately for every modeling language. If the enterprise offers some generic rules, then those rules can be reused across modeling languages. One example of rules based on the location of the modeling element within an i* model is a dependency between two actors. The notion of dependency implies that one actor depends on the other for performing a task, accomplishing a goal, or acquiring a resource. In i* the dependency is represented by a link between the two actors. This link incorporates a dependum (goal, task, or resource). To clarify, consider the following scenario: a person depends on a bank for fulfilling his goal of obtaining a loan. This scenario is depicted in Figure 15 by modeling two actors: “Person” and “Bank”, with a goal dependency between them labeled with “Loan” as dependum. While drawing this dependum on the modeling canvas, the modeler will receive as suggestions all ESO concepts corresponding to UFO Relators relating the ESO concepts used to annotate the two i* actors in the same model.

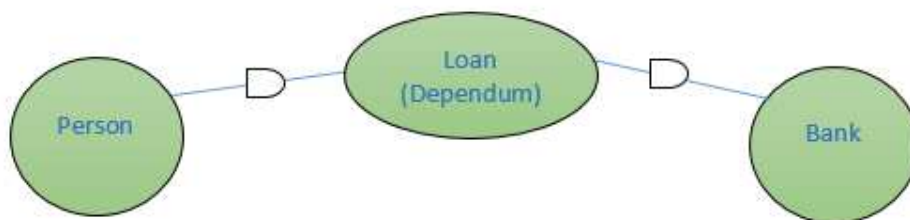


Figure 15: i* goal dependency

In order to implement the CMOE+ i* tool, a large part of the CMOE+BPMN tool was reused. In particular, the overall architecture and workflow was maintained, and the semantic annotation service (detailed in Section 3.2.4) and components implementing the recommendation services were re-used (yet using the i* specific models, i.e., ontological analysis of i*, the i*-specific recommendation rules). Furthermore, the user interface and general look and feel of the tool is identical to CMOE+BPMN. Hence, if modelers become acquainted with one tool, they will not have issues using any other CMOE+ tool. BPMN modeler was replaced with i* modeler. In order to choose the appropriate i* modeling tool,

the i* wiki²⁰ was consulted. Unfortunately, no up to date free of charge i* modeler was found, and therefore, some additional effort was required to align an existing tool with our needs. The Design CASE Tool for Agent-Oriented Repositories, Techniques, Environments, and Systems (Descartes) was selected as basis for CMOE+ incorporation, as it is written in Java, acts as a plugin for the Eclipse IDE and supports SD and SR models, all of which are compatible with the existing CMOE+BPMN implementation. However, Descartes was designed to work with the Tropos methodology, which is an older dialect of i*. Hence, as a first update to Descartes, it was necessary to incorporate the missing i* constructs and add the general look and feel of i*. The resulting CMOE+i* tool was configured to be able to access the ontology storage in order to grant the modeler access to the ontologies. Additionally, the recommendation services, and the semantic annotation mechanism were configured within the tool. Moreover, the tool converted the created i* models into OWL files and placed them into the model repository. Those models will be used in two different ways: First, for collecting feedback which can potentially result in ESO upgrade. Second, for enabling alignment with BPMN models which will be created in the future.

Figure 16 below presents a screenshot of the CMOE+i* tool. Caption 1 presents the constructs of the i* modeling language which the modeler will drag-and-drop to compose the i* model. Caption 2 highlights the ESO concepts presented to the modeler after he selects a modeling element from the i* model. ESO concepts appear as an ordered list, arranged based on the relevance score they receive from the recommendation services. The modeler selects any concept from the list (not necessarily from the top) and double-clicks it in order to annotate the selected modeling element with that ESO concept. In case the modeler did not find the appropriate ESO concept to be used for annotation, he can assign the modeling element a label of his choice. If no appropriate concept was found in the ESO, it is possibly an indication of the fact that this modeling element represents a missing concept in the ESO. The modeler can specify this element's label as *candidate annotation* (as presented in caption 3 of Figure 16). Labels marked as candidate annotation represent a special type of feedback, and are used to update the ESO if they are approved by the community in Community-based ontology feedback evaluation phase of CMOE+.

²⁰ http://istar.rwth-aachen.de/tiki-view_articles.php

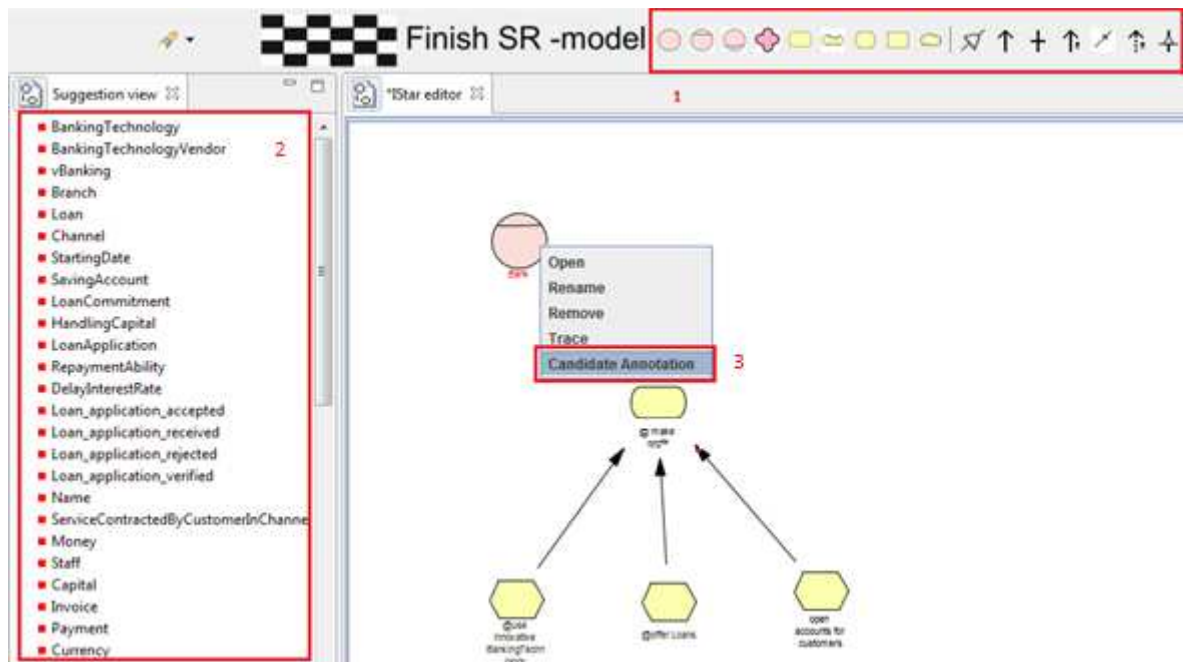


Figure 16: Screenshot of CMOE+i*

4.3 Extending CMOE+BPMN with I*-specific Recommendation Rules

In order to support the alignment between goal models and process models, previously described CMOE+ BPMN (Chapter 3) was extended with two features:

1. Extending the Ontology storage and recommendation generation phase of the Ontology Evolution Cycle of CMOE+ with mechanisms allowing storage and access of models previously created within the enterprise. The *model repository* (Section 4.2.1) is used to store all previously created models. Those models will be subsequently accessed and used to promote the alignment.
2. Extending the Rule-based recommendation service with rules specific to i* and BPMN modeling notations. This recommendation service gains access to models in the model repository and extracts the modeling elements which are the most relevant to the modeling element being created by the modeler. Those rules are explained in Section 4.2.2.

The same aspects need to be considered in order to enable the alignment among other modeling languages (other than i* and BPMN).

4.3.1 Model Repository

In the CMOE+ framework, when the modeler constructs a model using a particular modeling language, an OWL formalization of this model is created and stored in the model repository.

Figure 17 below presents an excerpt of a regular i* model created with the common i* syntax on the left, and the corresponding OWL representation. The purpose of storing the model in OWL format is the possibility to utilize the OWL reasoner to apply rules over these models. The recommendation services are formalized as SWRL rules which use the reasoner to query ontologies and models in OWL format. This OWL formalization contains all elements of the model and stores the ESO annotations. Only models created as a part of the CMOE+ framework are contained in the model repository; in order to utilize those models to the full potential, they must be annotated with ESO concepts, as the alignment between models is established through the ESO.

The model repository serves two purposes: first, models from the repository are used to extract information about possible changes in the knowledge of the enterprise. Any discovered changes are considered as feedback to the Ontology Evolution cycle. Second, models residing in the model repository are available for subsequent conceptual modeling cycles. In the context of the goal and process model alignment, previously created goal models are used by the recommendation services while creating process models.

Any type of conceptual model can be stored in the model repository. The tool is aware which type of models it is required to fetch, and thus the modeler is not expected to explicitly consult the model repository to find the potentially relevant models. The recommendation services are accessing the model repository automatically through the OWL API, as the models are stored in OWL format.



Figure 17: An excerpt of an i* model created by the CMOE+i* tool (on the left), and the corresponding OWL representation stored in the model repository (on the right)

4.3.2 Rule-Based Recommendation Services

The rule-based recommendation service of the CMOE+BPMN+ tool is extended to generate additional alignment recommendations for the modeler by merging all available ontologies

(ESO and OWL formalization of previously created i* models) and by taking into account a set of model alignment rules that were added to the CMOE+ framework. Next an ontology reasoner is used to generate a set of suggestions every time the model developer adds a specific element on the modeling canvas. The rule-based recommendation service is very flexible because different kinds of rules can be added. The rules can target the ESO, but can also focus on the meta model of the modeling language to allow alignment between constructs of different modeling languages. The rules that are added in this chapter all focus on the alignment between i* and BPMN models and have their origin in some of the previously mentioned alignment methods found in literature, and thus only operate on the i* and BPMN modeling notations. However, rules concerning other modeling languages can be easily configured, based on existing model alignment literature, or custom designed based on the needs of the enterprise. Next, the rules are formalized in SWRL (such as in Listing 4-7), and are added in the RuleO file (Section 3.2.3), which is an OWL file containing all the rules actively utilized by the Rule-based recommendation service.

Rule 1: i* Actor ➔ BPMN Pool

This rule is inspired by (Koliadis et al. 2006). Actors from the i* model are mapped to a pool in BPMN because this construct represents participants of the process. When the modeler creates a BPMN pool, all actors within i* models from the model repository are offered as suggestions for annotating the BPMN element. Listing 4 below shows the SWRL code for this rule.

```
i*:Actor(?actor), bpmn:Pool(?pool),  
sa:AnnotatedBy21(?actor, ?domain) -> sa:AnnotatedBy(?pool, ?domain)
```

Listing 4. Rule 1

Rule 2: i* Goal/Task ➔ BPMN Task

This rule appears frequently in works on goal and process model alignment (Decreus & Poels 2011; Guizzardi & Reis 2015; Santos et al. 2010; Koliadis et al. 2006). It maps an i* goal and task to the task construct in BPMN. This rule makes perfect sense as all those constructs mostly represent activities to be performed. Even though i* goals denote the state of affairs to be achieved, it is often labeled as an activity in the form of *verb*, *noun*. This rule implies that if a BPMN task is created, all i* tasks and goals from the model repository are derived as recommendation. The recommendations derived using this rule are rather broad, and will

²¹ The complete name of the object property is
 “SemanticAnnotation:representationClassIsAnnotatedByDomainClass”.

be further refined by other recommendation services. For example, if the modeler has entered a label for the modeling element, the Label-based recommendation service will compare the entered label with the suggested concepts using string and synonym matching. The SWRL code of this rule is presented in Listing 5 and 6 below.

```
bpmn:Task(?task), i*:Goal(?goal),  
  
sa:AnnotatedBy(?goal, ?domain) -> sa:AnnotatedBy(?task, ?domain)
```

Listing 5. Rule 2 (for i* goals)

```
bpmn:Task(?task1), i*:Task(?task2),  
  
sa:AnnotatedBy(?task2, ?domain) -> sa:AnnotatedBy(?task1, ?domain)
```

Listing 6. Rule 2 (for i* tasks)

Rule 3: i* OR Decomposition → BPMN Exclusive Gateway

This rule is inspired by (Santos et al. 2010). The exclusive gateway construct in BPMN represents a choice between alternative paths to be taken within the process. Identically, OR decomposition in i* reveals alternative activities to be performed to achieve the same goal. This rule is activated if the BPMN model contains an exclusive gateway, and there already exists one annotated task connected to this gateway. This rule is formalized in SWRL in Listing 7, and graphically represented in Figure 18. The recommendation service goes through the following steps to derive suggestions:

1. Check if there are i* models in the model repository containing tasks annotated with the same ESO concept as the annotation of the BPMN task
2. Inspect whether this annotated i* task is being a part of OR decomposition (means - ends) relationship
3. Verify if there are other tasks serving as means to the same end (goal) as the annotated task
4. Derive the tasks found in step 3 as suggestions

```

bpmn:Task(?btask1), bpmn:Task(?btask2), bpmn:Sequenceflow(?flow1),
bpmn:Sequenceflow(?flow2), sa:AnnotatedBy(?btask1, ?domain), bpmn:ExclusiveGateway(?gateway),
bpmn:hasTarget(?flow1, ?btask1), bpmn:hasSource(?flow1, ?gateway), bpmn:hasTarget(?flow2,
?btask2), bpmn:hasSource(?flow2, ?gateway), i*:ORrefinement(?or1), i*:ORrefinement(?or2),
i*:GoalTaskElement(?goaltask1), i*:GoalTaskElement(?goaltask2), i*:GoalTaskElement(?goaltask3),
i*:hasSourceRefinement(?or1, ?goaltask2), sa:AnnotatedBy(?goaltask2, ?domain),
i*:hasSourceRefinement(?or2, ?goaltask3), i*:hasTargetRefinement(?or1, ?goaltask1),

i*:hasTargetRefinement(?or2, ?goaltask1), sa:AnnotatedBy(?goaltask3, ?domain2) ->
sa:AnnotatedBy(?btask2, ?domain2)

```

Listing 7. Rule 3

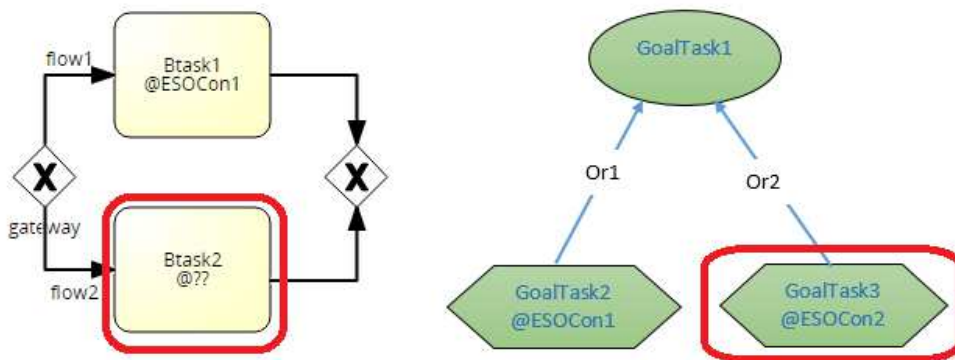


Figure 18: Graphical representation of Rule 3

Rule 4: i* AND Decomposition → BPMN Parallel Gateway

This rule, as its predecessor (Rule 3), is derived from (Santos et al. 2010). The BPMN Parallel Gateway construct shows activities that can be performed simultaneously. Subsequently, the i* AND decomposition presents all sub-activities to be accomplished in order to complete a particular task. This rule is activated if the BPMN model contains a parallel gateway, and if there is one annotated task connected to it. This rule is presented below in Listing 8, and graphically represented in Figure 19. The recommendation service goes through the following steps to derive suggestions:

1. Check if there are i* models in the model repository containing tasks or goals annotated with the same ESO concept as the annotation of the BPMN task
2. Verify whether this annotated i* task / goal is being a part of AND decomposition (task decomposition) relationship
3. Examine whether the annotated task / goal is in the right position within the AND decomposition relationship
4. Check if there are other tasks or goals participating in the same relationship at the same level as the annotated task / goal
5. Derive the tasks and goals found in step 4 as suggestions

```

bpmn:Task(?bpmntask1), bpmn:Task(?bpmntask2), bpmn:Sequenceflow(?flow1),
bpmn:Sequenceflow(?flow2), sa:AnnotatedBy(?bpmntask1, ?domain),
bpmn:ParallelGateway(?gateway), bpmn:hasTarget(?flow1, ?bpmntask1), bpmn:hasSource(?flow1,
?gateway), bpmn:hasTarget(?flow2, ?bpmntask2), bpmn:hasSource(?flow2, ?gateway),
i*:ANDrefinement(?and1), i*:ANDrefinement(?and2), i*:Task(?task),
i*:IntentionalElement(?element1), i*:IntentionalElement(?element2), i*:hasSourceRefinement(?and1,
?element1), sa:AnnotatedBy(?element1, ?domain), i*:hasSourceRefinement(?and2, ?element2),
i*:hasTargetRefinement(?and1, ?task),

i*:hasTargetRefinement(?and2, ?task), sa:AnnotatedBy(?element2, ?domain2) ->
sa:AnnotatedBy(?bpmntask2, ?domain2)

```

Listing 8. Rule 4

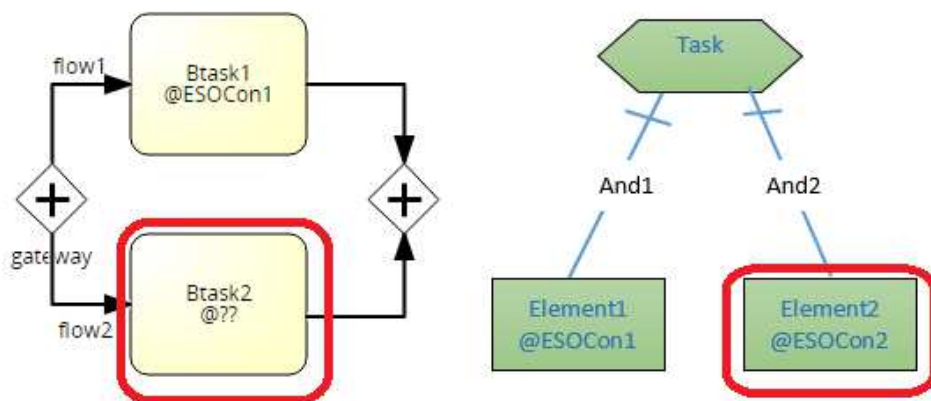


Figure 19: Graphical representation of Rule 4

Rule 5: i* Dependency → BPMN Message Flow

This rule is adapted from (Koliadis et al. 2006). Message flow in BPMN represents communication between two process participants (two BPMN pools). I* dependencies also signify interactions among different actors. Hence, there is a considerable chance that message flow between two pools in a BPMN model is representing one of i*'s dependencies between actors annotated with the same ESO concepts as the two pools. Rule 5 applies if there are two annotated BPMN pools and the modeler adds a message flow between them. The message flow can originate either from a pool or from a task within that pool. The formalization of this rule is presented in Listing 9, and graphically represented in Figure 20. Below are the steps followed by the recommendation service to derive suggestions based on this rule:

1. Verify whether there are models in the model repository containing actors annotated with the same ESO concepts as the two BPMN pools
2. Look for any dependencies (task/goal/resource) present between the two actors from step 1
3. Derive the dependencies as suggestions

```

bpmn:Pool(?pool1), bpmn:Pool(?pool2), bpmn:MessageFlow(?smsflow), sa:AnnotatedBy(?pool1,
?domain1), sa:AnnotatedBy(?pool2, ?domain2), bpmn:Task(?bpmntask),
bpmn:isLocatedWithin(?bpmntask, ?pool1), bpmn:hasSource(?smsflow, ?bpmntask),
bpmn:hasTarget(?smsflow, ?pool2), i*:Dependency(?dep1), i*:Actor(?actor1), i*:Actor(?actor2),
i*:IntentionalElement(?element), i*:dependee(?dep1, ?actor1), i*:dependee(?dep1, ?actor2),
i*:dependum(?dep1, ?element), sa:AnnotatedBy(?actor1, ?domain1), sa:AnnotatedBy(?actor2, ?domain2),
sa:AnnotatedBy(?element, ?domain3) -> sa:AnnotatedBy(?smsflow, ?domain3)

```

Listing 9. Rule 5

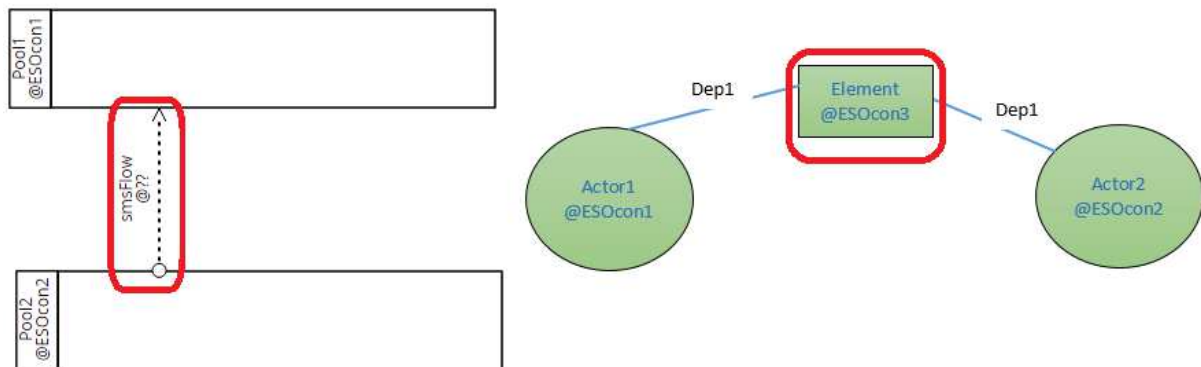


Figure 20: Graphical representation of Rule 5

4.4 Demonstration

In order to demonstrate the proposed method, a case about a hospital's emergency department is used. The case is inspired by a real-life emergency department in a local hospital, but it is simplified to facilitate a comprehensive demonstration and explanation of the proposal CMOE+ based i*-BPMN alignment. For the purpose of this case, it is assumed that the i* models representing the actors within the emergency department, their goals and dependencies are already created and stored in the model repository accessible by CMOE+BPMN tool. For this demonstration, an existing medical ontology was revised by an ontology engineer to more closely represent the ESO for an emergency department. The feedback from i* models was also incorporated in the ESO following the Ontology Evolution phase of CMOE+. The final ontology product was formalized in OWL and made accessible within the CMOE+BPMN tool.

The process model to be created is presented in Figure 21. When a patient arrives to the emergency room, the triage nurse obtains the patient's vital signs (blood pressure, temperature, and heartbeat) to estimate her condition. Afterwards, the emergency physician attempts to determine the patient's illness by viewing her present symptoms and past medical history obtained from patient's primary care provider. The patient needs to communicate all the experienced symptoms as detailed as possible. This helps the physician

to produce differential diagnoses. The most likely diagnoses are then determined by physical examination. When the patient is diagnosed, she receives the right treatment if it is possible in the emergency hospital. If the diagnoses are determined, but the treatment is not offered by the emergency department, the patient is redirected to a more specialized medical facility.

While adding elements to the modeling canvas, the modeler receives suggestions derived from the ESO and the model repository. Those suggestions aim to facilitate the modelling process by showing potentially useful or forgotten concepts (that are present in goal models previously created) and encouraging the usage of consistent vocabulary. The modeler is free to adopt these suggestions, or disregard them and look further in the ESO for more suitable concepts, or in case none found, to introduce a custom label. It is important to note that the suggestions are concepts with a higher probability to fit a particular BPMN construct.

This section aims to demonstrate the type of suggestions derived based on the five rules described in Section 4.3.2 while creating the process model in Figure 21, and what are the potential benefits of those suggestions. Figure 21 highlights in red the areas corresponding to each of the five rules. Every highlighted area is numbered according to the rule it follows.

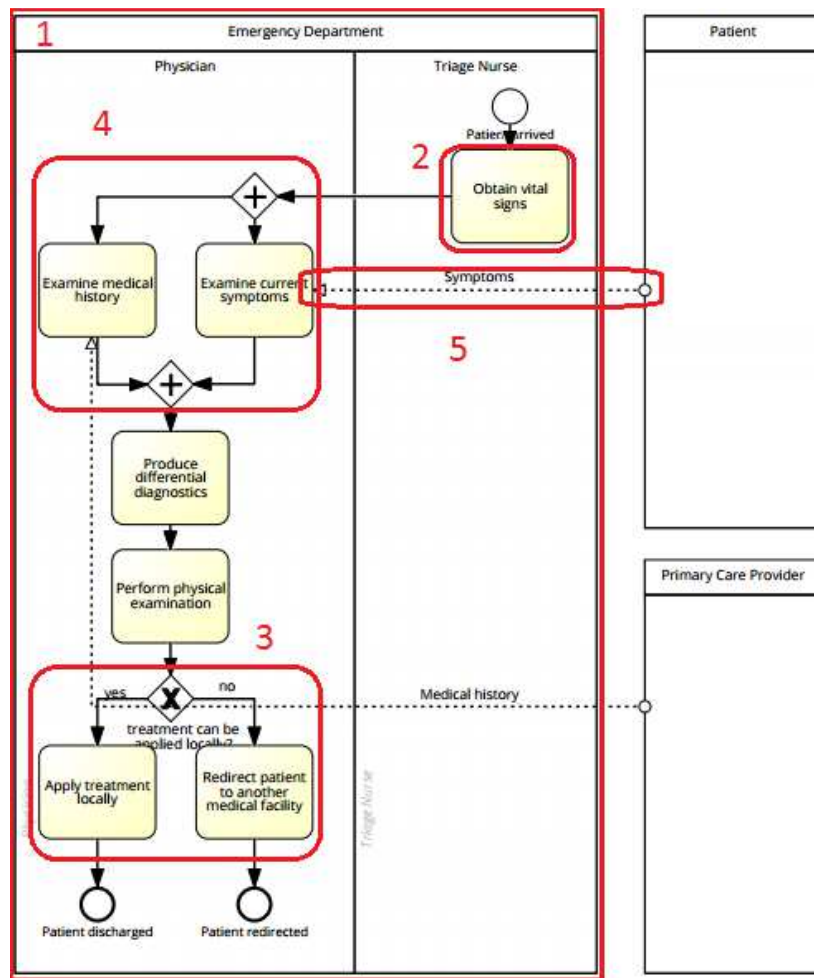


Figure 21: Simplified process model for the emergency department

RULE 1: While placing a pool on the modeling canvas, all the ESO concepts corresponding to i* actors are derived as suggestions. It is important to mention that this is only a handful of concepts compared to the full capacity of ESO. This facilitates the ontology lookup process greatly, and allows the modeler to potentially find the relevant concept rapidly, and possibly to discover and add actors she did not think of. The limited size of the set of actors also allows a fast and easy comparison between the pools in the process model, and the actors in i* models.

RULE 2: the majority of concepts in the ESO used for this demonstration are nouns. This rule brings forward the few ESO concepts with *verb noun* structure. This rule promotes the usage of a correct form of labeling for a BPMN task (verb noun) (Dumas et al. 2013).

RULE 3: suggests the ESO concepts corresponding to i* tasks which are specified as alternative means to achieve the same goal. This rule presents to the modeler all the alternative paths possible in a particular situation. This might inspire the modeler to consider alternatives she has not thought of in the process model. This rule limits the suggestion list significantly as it only suggests tasks used to achieve a particular goal.

Furthermore, Rule 3 facilitates detecting incompleteness in i* models; the number of the suggested concepts is very limited (typically two or three concepts in the described case) which does not require much effort from the modeler to spot any missing concepts.

RULE 4: offers the same benefits as Rule 3 while operating based on i* Task Decomposition link rather than i* Means – Ends link.

RULE5: proposes as suggestions all the possible interactions between the i* actors corresponding to the BPMN pools (Emergency Department and Patient in Figure 21). Those interactions are represented in i* models by means of dependencies where one actor depends on the other for achieving a goal, performing a task, or acquiring a resource. ESO concepts used to annotate those i* goals, tasks, and resources will be derived as suggestions. This rule is less specific than Rule 3 and 4. However, it can inspire the modeler to add interactions which were disregarded in the process description.

4.5 CMOE+X Tool Architecture

CMOE+ is architected in a manner that it is easy to instantiate, and only a few components need to be changed for every instantiation. The remaining components remain fixed. Figure 22 represents a component diagram of the CMOE+ tool. The User Interface Layer interacts with the user using three main components: the first component, the *Recommendations view*, is responsible for displaying the ordered list of ESO concepts. The *ESO properties view* presents properties of different ESO concepts. Those properties include the definition, the individual relevance scores assigned by each recommendation service, and a weighted average of those scores. The *Model editor* captures modeling elements selected by the modeler. The communication between the modeler and the view is facilitated by the *Adapters* from the Business Logic Layer. The views and the model editor are acting as interface between the modeler and the code. Only the Model editor component needs to be replaced when CMOE+ is configured with a different modeling language. The Business Logic Layer features two main components: the *Recommendation services* and the *Adapters*. The *Recommendation services* component is the Java implementation of all the active recommendation services. It receives information on the modeling element selected by the modeler, consults the *ontology storage* and/or *model repository* via the *ontology manager*, calculates the relevance scores per every services, and the weighted average, and finally passes the resulting information to the views in User Interface Layer. The *Recommendation services* component is altered for every modeling language because some of the recommendation services are specific to the modeling language, and hence cannot be reused across languages. The Data Access Layer is responsible for transmitting information about the database to the Recommendation services. It uses the OWL API to access the ontologies and models represented in OWL format. The *Ontology storage* database is updated with every modeling language, as some ontologies residing within are dealing

directly with the modeling language at hand, such as the ontology of the meta-model of the modeling language, and ontological analyses of the modeling language. The Model repository is constantly updated with new models. However, such updates are independent of the modeling language, and the models are added to the repository automatically.

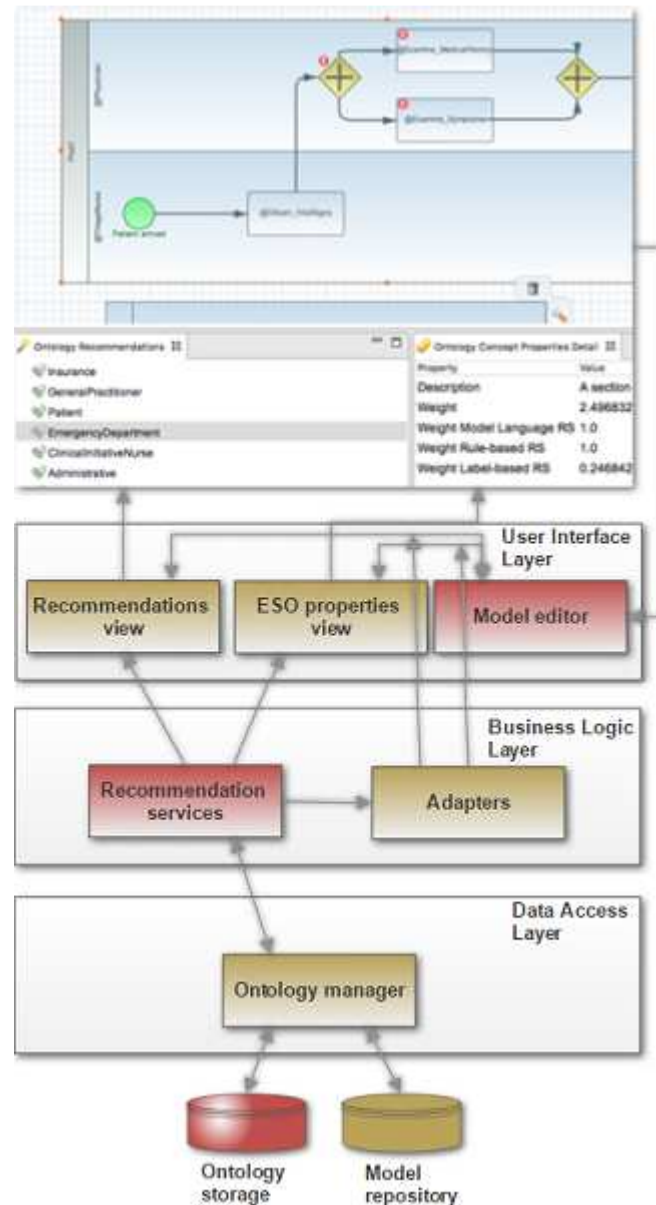


Figure 22: Components of CMOE+ tool. The components highlighted in red need to be altered for every modeling language

4.6 Conclusion

This chapter starts by introducing the literature on goal and process model alignment. Next, it demonstrates how CMOE+ contributes to establishing the alignment between i* and BPMN models, and which extensions to CMOE+ were required to support this. Additionally,

this chapter presents a new instantiation of CMOE+ with the i* modeling language. This chapter also iterates on the steps to be performed to configure the CMOE+ tool with a new modeling language.

With every new instantiation of CMOE+, the original framework is analyzed and updated. Those updates, even if minor, are still important to increase the quality of CMOE+. The final version of the Conceptual Modeling and Ontology Evolution framework by the end of this dissertation, is presented in Figure 23 below. By looking to Figure 23, the reader will notice that the model repository was added to the *Ontology and model storage* phase (the name of the phase has also been changed to accommodate the addition). Moreover, the recommendation services were removed from within the phase, and placed between the two cycles. This was motivated by the fact that the recommendation services are not a part of a particular phase, but are operating between phases (and even cycles). It is important to mention that CMOE+ has not stopped evolving, and additional enhancements are foreseen based on ongoing research.

Ontology Evolution cycle

Conceptual Modeling cycle

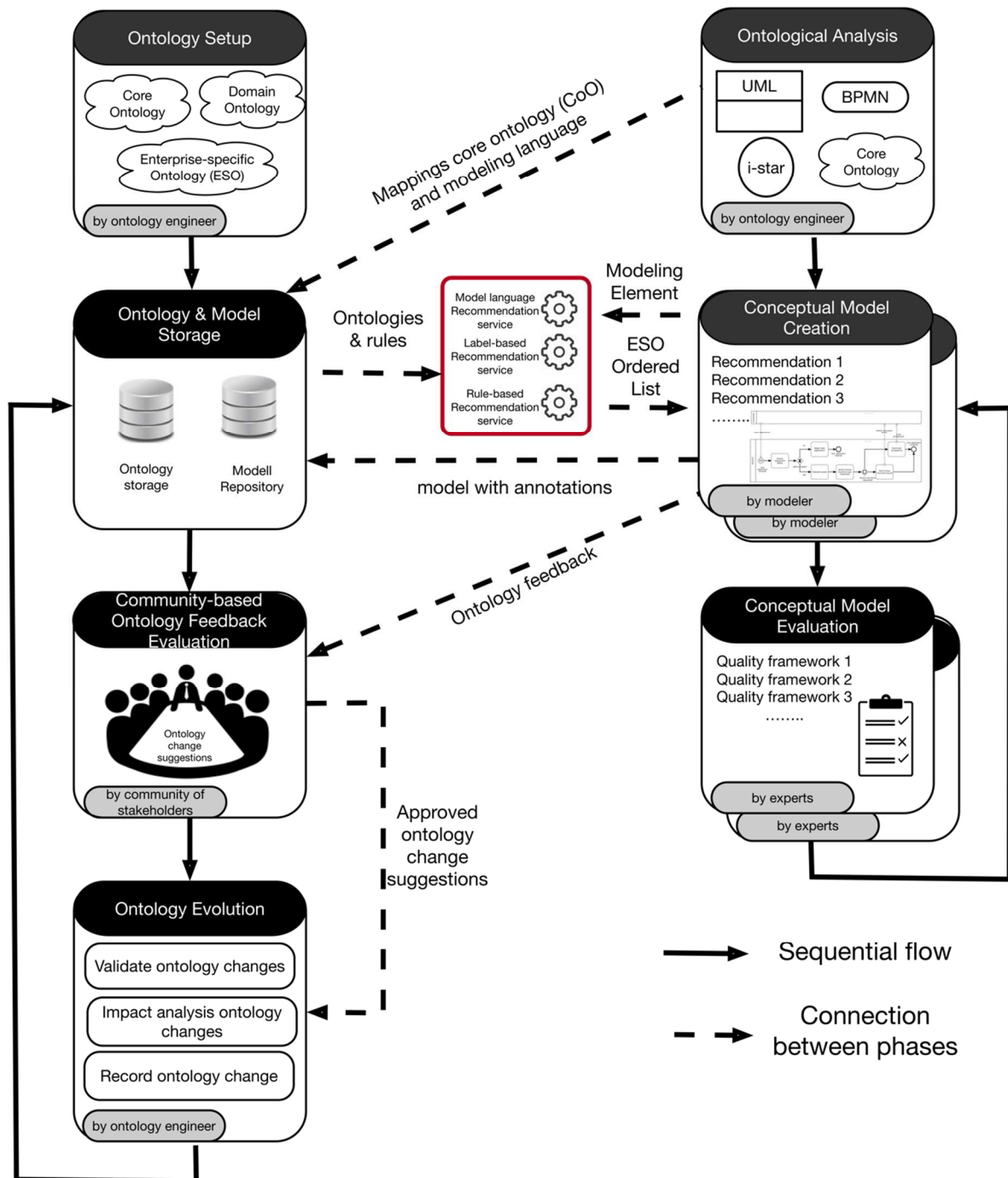


Figure 23: The final version of CMOE+

5

Conclusions and Future Work

This chapter discusses the results of this work, elaborates on the contributions, highlights some limitations, and gives directions for future work.

5.1 Research Results

The main purpose of this research work is to support the modeler in creating semantically aligned conceptual models based on an enterprise specific ontology. In order to accomplish this, the Conceptual Modeling and Ontology Evolution framework (CMOE+) was developed, which consists of two cycles, the Conceptual Modeling and the Ontology Evolution cycle, and establishes a symbiotic relationship between them. In this thesis, the CMOE+ framework was outlined as a whole, but the focus lies on the conceptual modeling phase, which is described and evaluated in detail; the ontology evolution phase is not fully explored, and only described schematically.

The conceptual modeling cycle of CMOE+ facilitates the creation of semantically aligned conceptual models. To do so, some preparatory work is required: preparing and setting up the various conceptual modeling languages used within the enterprise. Once set up, in the ontology engineering cycle, the modeler is guided and assisted to create semantically annotated models, based on an enterprise-specific ontology. As such, CMOE+ support the main goal of this thesis. Finally, to ensure high quality models, the conceptual modeling cycle prescribes a quality check on the resulted conceptual model.

On the other hand, the *Ontology Evolution* cycle concerns the ontological support for the conceptual modeling cycle. It facilitates ontology storage, offers support to establish the

initial version of the ESO, and makes it accessible in a convenient way to various other processes operating within CMOE+, most importantly the recommendation services, which offer modeling recommendations during the conceptual modeling process (in the conceptual modeling phase). Finally, this cycle is also responsible to process feedback from the conceptual modeling cycle, and utilize it to update and evolve the enterprise specific ontology, in order to better match the evolving needs of the enterprise.

The CMOE+ framework offers guidelines for every stakeholder involved in the conceptual modeling or ontology engineering practices. Simultaneously, the framework is flexible and easy to accommodate to the specific needs of enterprises. This flexibility includes configuring new modeling languages, quality evaluation frameworks, adding recommendation services, and other changes. Concerning the Ontology Evolution cycle, the Ontology setup phase needs to be performed only once to acquire the initial version of ESO. The CMOE+ framework offers several possibilities within this phase, and it is up to the ontology engineer to make a decision on how the initial ESO is established. The second phase, Ontology storage and recommendation services, provides an opportunity to extend the recommendation services with new matching algorithms corresponding to the alignment rules used by the enterprise, or new matching algorithms to accommodate newly added modeling languages (as some recommendation services are language specific). Community-based ontology feedback evolution allows incorporating any community-based interaction fora which an enterprise finds appropriate. There are several approaches in the literature which do not require the community members to be present at the same location (as presented in Section 2.3.1.3), which is convenient for enterprises with physically distributed stakeholders. The ontology evolution phase leaves the choice up to the ontology engineer on how often to update and evolve the ESO, and how to tackle versioning issues.

The Conceptual Modeling cycle also keeps the possibility open for changes and updates. It's first phase, Ontological analyses of the modeling language, allows incorporating new modeling languages into CMOE+. The Conceptual model creation phase offers an interface to the ESO during model creation, and is responsible for the semantic annotation. This is the most rigid phase within CMOE+, and most of the actions performed here are predefined. However, it is still possible to adjust the way the ESO concepts are presented to the modeler, if there is a need for that. The final phase, Conceptual model evaluation, allows incorporating quality frameworks to assess the quality of the created models. Some of the quality criteria are enforced during model creation. For example, syntactic quality can be enforced by the modeling tool, as nowadays many modeling tools incorporate a built-in quality check. Semantic quality is also re-enforced to some degree by basing the models on the ESO (which represents the agreed upon knowledge of the enterprise). Hence, this phase is mostly concerned with quality evaluation of the end product.

In addition to establishing guidelines for CMOE+, within this work we have presented a concrete instantiation of the framework for BPMN, the CMOE+BPMN framework, and

implemented the CMOE+BPMN tool. CMOE+BPMN was described and evaluated in detail in Chapter 3. In Chapter 4, a second instantiation of CMOE+, for the i* modeling language, was briefly reported, and the ability of CMOE+ to contribute to the semantic alignment of different modeling languages (i* and BPMN) was presented in Chapter 4. This demonstration provides evidence that CMOE+ is indeed capable of accommodating different modeling languages, and shows the overall flexibility of the framework. In the process of developing CMOE+, a set of recommendation services was put forward. The recommendation services cooperate on scanning the ontology and calculating the relevance score for every ontological concept. Those recommendation services can be reused independently of CMOE+.

In the beginning of this dissertation, Section 1.4 highlighted three research goals to be achieved within this research. To add structure to this section, and to connect it with the introduction, the obtained research results are now presented per research goal.

GOAL1: Setting overall requirements for establishing the symbiotic relationship between conceptual modeling and ontology engineering, and establishing a framework which will deliver those requirements.

In order to accomplish this goal, we took part in the FRIS case and reviewed relevant literature. As a result, seven requirements were put forward (Section 2.2). Next, the structure of CMOE+ emerged in accordance with these requirements. The framework was divided into two cycles and various phases per cycle. The description of every phase was supplied in Section 2.3. CMOE+ possesses the following characteristics:

- CMOE+ is a general framework which is not bound to a particular modeling language or modeling paradigm. CMOE+ is flexible, and every phase is configurable.
- It promotes semantic alignment during model creation
- The framework establishes semantic alignment of models by means of semantic annotation which guarantees the modeler the freedom of choosing the terminology for labeling the modeling elements.

GOAL2: Providing more detailed explanation regarding those phases of the framework which are related to the conceptual modeling activity.

After offering a brief description of every phase in relation to GOAL1, a detailed explanation of phases related to the conceptual modeling part was provided in Section 2.3. This dissertation focuses on the conceptual modeling activity, hence some phases of the Ontology Evolution cycle are not elaborated and remain future work. Figure 24 highlights the phases explored within this dissertation.

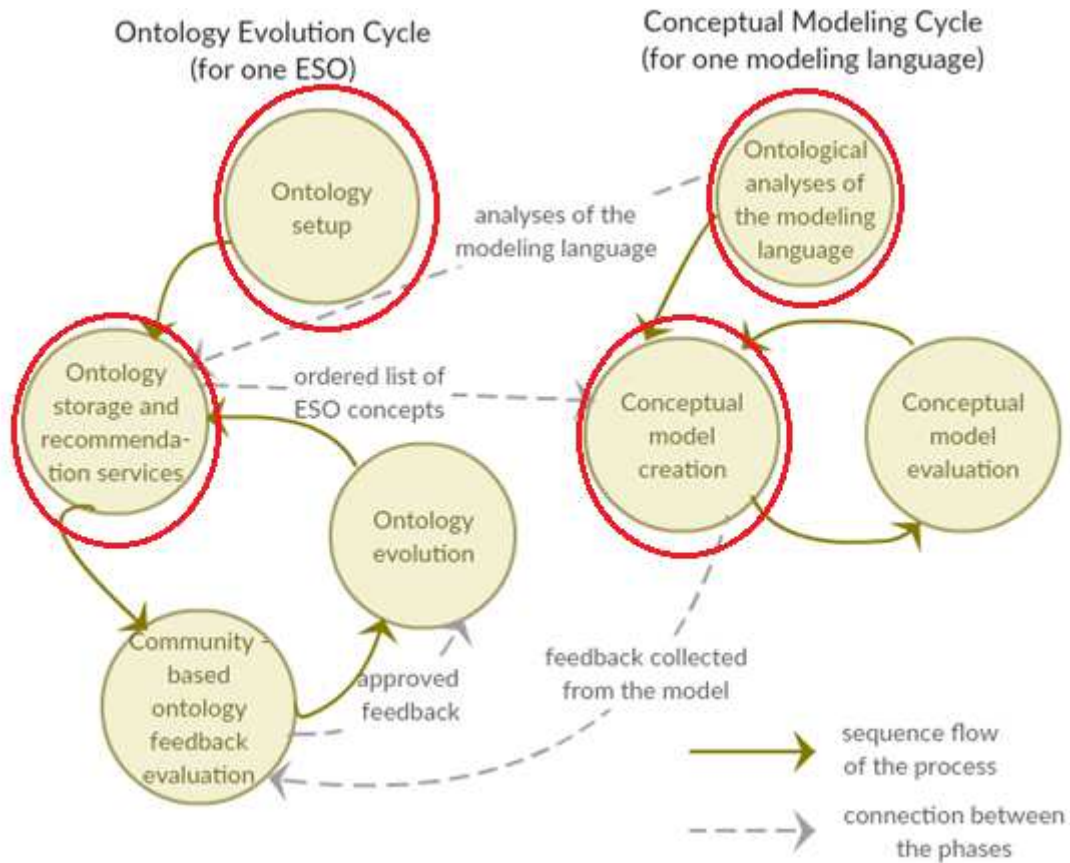


Figure 24: Phases of CMOE+ which are elaborated in detail within this dissertation

After elaborating on the conceptual modeling phases of CMOE+, the resulting framework exhibits the following characteristics:

1. *Promoting semantic alignment of models using ESO.* Within CMOE+, creating conceptual models based on ESO is highly encouraged by displaying ESO concepts to the modeler during model creation. The modeler annotates each modeling element with one ESO concept, thereby establishing a semantic connection between the ESO and the conceptual model. The modelers maintain the freedom of choosing the appropriate label for the modeling elements, and is not restricted by the terminology presented by ESO concepts.
2. *Encapsulating the ESO and presenting it in a manner which does not require any specialized knowledge on ontologies from the modeler.* During the modeling process, the modeler accesses the ESO through an interface which conceals all the technical aspects of the ontology, and the modeler is only presented with an ordered list of ESO concepts. The modeling task using our approach was tested with students who do not have knowledge on ontologies. As the reader can see from Chapter 3, the lack of knowledge did not hinder the usability of our approach. However, for ontology-related issues such as establishing the initial version of the ESO, and maintaining it up to date, the interference of the ontology engineer will be required.

3. *The semantic alignment is promoted during model creation.* While creating the conceptual model, the modeler uses the annotation mechanism to link every modeling element (if possible) to the corresponding ESO concept. This ensures that the model is based on the ESO already during the creation, and no further intervention is required on the final product.

Ability to suggest the most relevant ESO concepts to the modeler. CMOE+ incorporates several recommendation services which access the ESO and scan the ontological concepts extracting the most relevant ones to be suggested to the modeler for annotating the modeling elements. The recommendation services constitute of rules and algorithms calculating the relevance score for each ESO concept and presenting an ordered list of ESO concepts to the modeler. The recommendation services are very flexible, and can be extended to correspond better to the needs of the enterprise, and the modeling languages used.

GOAL3: Evaluating the phases responsible for the conceptual modeling side of the symbiosis by demonstrating the framework for BPMN process modeling language, and i* goal modeling language.

In order to accomplish this goal, different phases of CMOE+ were instantiated for BPMN. This required analyzing BPMN modeling language with UFO core ontology, and incorporating the relevant recommendation services. Next, CMOE+BPMN prototype was implemented using Java in the Eclipse environment. The tool was evaluated in an experiment with students. In less detail, the elaboration of CMOE+ for i*, CMOE+i*, and the accompanying tool, were also described.

GOAL4: Exploring the ability of CMOE+ in aligning i* goal models with BPMN process models.

The purpose of this PhD is to facilitate creation of more semantically aligned conceptual models. Hence, it was important to test the proposed alignment method using two different modeling languages, as the underlying premise is that semantically annotating different conceptual models with the same ESO, will promote mutual semantic alignment. Chapter 4 presents the idea of the alignment and the modifications required for CMOE+ to accommodate new modeling languages. Additionally, Chapter 4 demonstrates a second way in which CMOE+ can be used to align conceptual models created in different modeling languages: incorporating recommendation rules which draw from already annotated, previously created models. This was illustrated using i* and BPMN.

5.2 Contributions

This thesis combines reusing literature and proposing innovative ideas. The work performed contributes both to the research field, and to practice. While developing CMOE+, several iterations were made until the current version was established. Those iterations resulted in ideas, guidelines, methods, and other artefacts which can be utilized in practice. Some of those artefacts are bound to CMOE+, while others can be utilized independently. This includes the following practical contributions:

- *Proposing the Conceptual Modeling and Ontology Evolution framework.* The CMOE+ framework itself is the major contribution of this thesis. It combines aspects from various fields such as modeling, ontology engineering, ontology matching, semantic annotation, etc. to offer a comprehensive solution for the creation of semantically aligned models, and maintenance of the ESO. CMOE+ is flexible enough to operate with different modeling languages, and to allow the accommodation of the enterprise's needs. This makes the framework easy to tailor to different enterprises. Next to introducing the overall CMOE+ framework and describing it, this research elaborates the phases of CMOE+ related to conceptual modeling, presents its architecture, and highlights what needs to be altered in order to accommodate new modeling languages.
- *Developing a set of recommendation services* which can effectively perform ontology lookup and suggest to the modeler the ESO concepts which are the most relevant to the modeling elements being created. Some of the presented services are modelling language specific, others are transversally usable. The recommendation services can be reused outside the context of CMOE+ as new rules and algorithms are easily configured.
- *Establishing a method for semantic annotation of conceptual models.* In the process of creating conceptual models, the modeler can easily annotate elements of his models with ESO concepts. This annotation plays an essential role in enforcing semantic alignment between models, while simultaneously allowing a free choice of terminology. The mechanisms presented to semantically annotate conceptual models are equally usable outside the context of CMOE+.
- *Implementing a prototype of the CMOE+BPMN tool.* This tool incorporates all the features of CMOE+ (which were addressed in this dissertation). This tool exemplifies CMOE+ for process modeling using BPMN. While explaining the tool, this thesis is offering guidelines on how to instantiate CMOE+ for other modeling languages. We also briefly elaborated the CMOE+i* tool, which supports the i* modelling language.

- *Introducing a method for establishing alignment between models created using different modeling languages.* CMOE+ incorporates features allowing to align newly created models with models residing in the model repository. This was demonstrated while aligning BPMN models (while they are being created) with i* goal models from the model repository.

Contributions to the research field are summarized as following:

- *Applying ontological analysis of the modeling languages in practice.* A considerable amount of research is found on ontological analysis of modeling languages such as BPMN, i*, UML, etc. However, to the best of our knowledge, those research efforts remain theoretical where the researchers stop at establishing those analyses. Within the context of CMOE+ framework, ontological analysis from the literature can now be applied in practice. Until this point, CMOE+ was tested with ontological analyses of i* and BPMN using UFO foundational ontology.
- *Utilizing the alignment rules in practice.* Similar to the contribution above, this research work applies previously established rules aligning i* and BPMN models. Moreover, there is a possibility to incorporate rules aligning models in other modeling languages. Hence, CMOE+ offers a possibility to test a wide range of different alignment rules existing in literature.
- *Introducing the symbiotic relationship between conceptual modeling and ESO evolution.* Using the ESO to increase model alignment is not a new concept by itself. However, utilizing the created conceptual models as means to gather feedback about the enterprise and maintain the ESO, is more of an innovative idea. This idea is not fully explored within the context of this research work, but sufficient guidelines are offered in order to comprehend the idea conveyed.

5.3 Limitations and Future Work

Despite the effort that was invested in this thesis, this research is prone to a few limitations. First, it was only tested with two modeling languages. This can be regarded as a limitation because it poses a threat to the generalizability of CMOE+. However, the framework performed well on both modeling languages, and we identified the aspects that needs to be taken into consideration when a new modeling language is configured. Additionally, it is demonstrated how alignment between models in both modeling languages can be achieved, which shows the usability of CMOE+ across modeling languages. The second limitation is concerning the evaluation. The exemplification of CMOE+ with BPMN was tested with students. Even though the students are not the targeted population for CMOE+, students

with appropriate background and knowledge are often perceived as valid proxies for the practitioners. Finally, this work is lacking a comprehensive evaluation of the proposed framework. This unfortunately was not possible to achieve due to the fact that CMOE+ is not completed yet. As the reader can see, there are a few limitations. However, those limitations can be addressed in the future.

This research project will continue, and we will gradually keep improving CMOE+. The first part of the future work addresses the phases of CMOE+ which were completed within this dissertation:

1. There is a possibility to look further into the *Ontological analyses of the modeling language*, and the *Ontology storage and recommendation services* phases. Within the former phase, it is feasible to perform (or search for) ontological analyses of commonly utilized modeling languages using UFO. UFO is recommended as a foundational ontology within CMOE+. Hence, it would be beneficial to have such ontological analyses at our disposal. For the Ontology storage phase, it is planned to look into the recommendation services to find an optimal combination of algorithms, or add new algorithms to improve the suggested ESO concepts received by the modeler for every modeling element.
2. In addition to improving the individual phases, we are very curious *to test CMOE+ with practitioners and hear their opinion on the framework*. Even though students are often participating in research experiments, it is important to evaluate the Conceptual Modeling cycle of CMOE+ with real practitioners. They will be able to offer valuable insights on how to improve CMOE+ to suite organizational needs for modeling. This is something the students are not capable of doing. The feedback acquired from practitioners can also result in altering the Ontology Evolution cycle.
3. Another important aspect to be investigated is *establishing alignment among modeling languages* other than i* and BPMN.

The second part of the future work concerns the remaining phases of the Ontology Evolution cycle, which deal with gathering and validating the updated knowledge within the enterprise, and incorporating it into the ESO. Within the context of this dissertation, only the general description was provided for the Community-based ontology feedback verification phase, and Ontology evolution phase. Those phases constitute the biggest part of the future work. There is an intention in the research group to acquire a PhD student who will work on this topic. The following objectives are already highlighted in that regard:

1. *Facilitating gathering the knowledge on what has evolved within the enterprise*. In order to be able to capture the updates within the enterprise, CMOE+ suggests analyzing conceptual models created within this enterprise. All the models created

using CMOE+ are stored in the model repository. Those files are used to search for feedback. One way to search for feedback is to use an automated tool which looks for predefined patterns in model files. However, this does not allow capturing changes instantly after their occurrence. The changes need to be reflected in a conceptual model first.

2. *Validating the gathered knowledge (feedback).* After gathering the feedback, the next step is to validate whether this feedback is worth incorporating into the ESO. Even if some patterns were discovered in the previous step, they are not necessarily reflecting true changes to the enterprise. They can be representing some very localized changes, or maybe just a mistake made by the modeler. Therefore, the feedback is required to be carefully evaluated before it is used for ESO update. This evaluation is done by the certified members of the enterprise's community. CMOE+ foresees an online community forum where community members can vote upon the presented feedback.
3. *Tackling issues regarding ontology update* and ontology versioning, and the procedure of updating artefacts already connected to the previous versions of the ontology.

After completing all the phases of CMOE+, we have the following plans for the framework:

1. *Investigating how CMOE+ can be utilized to improve Business/IT alignment* within companies. ESO plays an important role in Business/IT alignment, and in improving the communication across the enterprise generally. Hence, after making progress on the Ontology Evolution cycle of CMOE+, it is worth investigating the impact of CMOE+ on the Business/IT alignment. Additionally, it is planned to test CMOE+ with existing Business/IT alignment approaches such as Archimate (CMOE+Archimate).
2. *Identifying and implementing the non-functional requirements of CMOE+* is another important aspect. Specially when the framework is intended for a practical usage. This thesis mainly focused on the functional requirements of CMOE+.
3. When all the phases are completed, CMOE+ framework will be *evaluated as a whole*, including all the phases within the context of a real enterprise. Annually, our research groups organizes a practice oriented workshop, inviting all the members of our Industrial Liaison Initiative. During the workshop all the projects of the research group are presented and for every project it is pointed out how this project can be useful in practice, and how practice can be involved in the next steps of the project. Of course, this type of evaluation requires a complete implementation of CMOE+

including the non-functional requirements. Observing its overall performance will be very interesting and satisfying for us.

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
ARIS	ARchitecture for integrated Information Systems
BI	Business Intelligence
BPMN	Business Process Modeling Notation
CASE	Computer Aided Software Engineering
CMOE+	Conceptual Modeling and Ontology Evolution Framework
CoO	Core Ontology
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
EC	Engineering Cycle
ER	Entity Relationship
ERP	Enterprise Resource Planning
ESO	Enterprise-Specific Ontology
FRIS	Flanders Research Information Space
IU	Intention of Use
MEM	Method Evaluation Model
MLO	Modeling Language Ontology
MoO	Model Ontology
OIL	Ontology Inference Layer
OWL	Web Ontology Language
PEOU	Perceived Ease Of Use
PU	Perceived Usefulness
RC	Research Cycle
REA	Resource-Event-Agent
RDF	Resource Description Framework
RulesO	Rules Ontology
SBVR	Semantics of Business Vocabulary and Business Rules
SQ	Semantic Quality
SRML	Simple Rule Markup Language
SUMO	Suggested Upper Merged Ontology
SWRL	Semantic Web Rule Language
TOVE	Toronto Virtual Enterprise Ontology
UEML	Unified Enterprise Modeling Language
UFO	Unified Foundational Ontology
UML	Unified Modeling Language
XML	Extensible Markup Language

References

- Abramowicz, W. et al., 2007a. Semantically enhanced Business Process Modeling Notation. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*.
- Abramowicz, W. et al., 2007b. Semantically enhanced Business Process Modelling Notation. *CEUR Workshop Proceedings*, 251.
- Aguilar-Savén, R.S., 2004. Business process modelling: Review and framework. *International Journal of Production Economics*, 90(2), pp.129–149.
- Allweyer, T., 2016. *BPMN 2.0: Introduction to the standard for business process modelling*, BoD - Books on Demand.
- Almeida, J.P. & Guizzardi, G., 2013. An ontological analysis of the notion of community in the RM-ODP enterprise language. *Computer Standards & Interfaces*, 35(3), pp.257–268.
- Aßmann, U. & Zschaler, S., 2006. Ontologies, meta-models, and model-driven paradigm. In C. Calero, F. Ruiz, & M. Piattini, eds. *Ontologies for Software Engineering and Software Technology*. Springer Berlin Heidelberg, pp. 249–273.
- Auer, S., 2006. RapidOWL - an Agile Knowledge Engineering Methodology. In I. Virbitskaite & A. Voronkov, eds. *6th International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer Berlin Heidelberg, pp. 424–430.
- Ayad, S., 2012. A Quality Based Approach for the Analysis and Design of Business Process Models. In *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*. Valencia: IEEE, pp. 1–5.
- Barcellos, M.P., Falbo, R.D.A. & Dal Moro, R., 2010. A well-founded software measurement ontology. *Frontiers in Artificial Intelligence and Applications*, pp.213–226.
- Bechhofer, S. et al., 2003. OWL web ontology language reference.
- Becker, J., Breuker, D., et al., 2009. Constructing comparable business process models with domain specific languages - an empirical evaluation. In *SCIS 2009 proceedings*.
- Becker, J. et al., 2009. Constructing comparable business process models with domain specific languages – an empirical evaluation. *Proceedings of the 17th European Conference on Information Systems (ECIS)*, pp.1–13.
- Becker, J. et al., 2010. Semantic Business Process Management Handbook on Business Process Management 1. In J. vom Brocke & M. Rosemann, eds. Springer Berlin Heidelberg, pp. 187–211.
- Becker, J. et al., 2009. Towards increased comparability of conceptual models-enforcing naming conventions through domain thesauri and linguistic grammars. *ECIS 2009 Proceedings*, pp.1–13.
- Becker, J., Delfmann, P., et al., 2009. Towards increased comparability of conceptual models - enforcing naming conventions through domain thesauri and linguistic grammars. In *ECIS 2009 proceedings*.
- Becker, J., Rosemann, M. & von Uthmann, C., 2000. Guidelines of Business Process Modeling. *Lecture Notes in Computer Science*, 1806, pp.30–49.
- Bera, P., Burton-Jones, A. & Wand, Y., 2011. GUIDELINES FOR DESIGNING VISUAL ONTOLOGIES TO SUPPORT KNOWLEDGE IDENTIFICATION. *Mis Quarterly*, 35(4), pp.883–908.
- Bera, Burton-Jones, A. & Wang, Y., 2011. Guidelines for designing visual ontologies to support knowledge identification. *Management information systems quarterly*, 35(4), pp.883–908.
- Blomqvist, E., 2005. Fully automatic construction of enterprise ontologies using design patterns: Initial method and first experiences. *Lecture Notes in Computer Science*, 3761(On the Move to Meaningful Internet Systems 2005: COOPIS, DOA, and ODBASE, Pt 2), pp.1314–1329.

- Bommel, P. van & Hoppenbrouwers, S., 2007. QoMo: A modeling process quality framework based on SEQUAL. In *Proceedings of EMMSAD*.
- Booch, G., 1994. *Object-oriented Analysis and Design with Applications*.
- Born, M., Dörr, F. & Weber, I., 2007. User-friendly semantic annotation in business process modeling. In M. Weske, M. Hasid, & C. Godart, eds. *Web Information Systems Engineering–WISE 2007 Workshops*. Nancy, France: Springer Berlin Heidelberg, pp. 260–271.
- Cabral, L., Norton, B. & Domingue, J., 2009. The Business Process Modelling Ontology. In *Proceedings of the 4th International Workshop on Semantic Business Process Management*. SBPM '09. New York, NY, USA: ACM, pp. 9–16.
- Cardoso, E., Guizzardi, R.S.S. & Almeida, J.P.A., 2011. Aligning goal analysis and business process modelling: a case study in healthcare. *International Journal of Business Process Integration and Management*, 5(2), pp.1–15.
- Chen, P., 1976. The Entity-Relationship model - toward a unified view of data. , 1(1), pp.9–36.
- Chow, L., 2011. BPMN 2.0 Primitives and Semantic Technology - Proof of Concept. *Talk presented on July 13, 2011*, (July).
- Commentary, R. et al., 2002. Research Commentary: Information Systems and Conceptual Modeling— A Research Agenda. *Information Systems*, 13(4), pp.363–376.
- Davies, I. et al., 2006. How do practitioners use conceptual modeling in practice? *Data and Knowledge Engineering*, 58(3), pp.358–380.
- Davis, F.D., 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *Mis Quarterly*, 13(3), pp.319–340.
- Debruyne, C. et al., 2011. Publishing Open Data and Services for the Flemish Research Information Space. In O. De Troyer et al., eds. *Advances in Conceptual Modeling. Recent Developments and New Directions*. Springer Berlin Heidelberg, pp. 389–394.
- Decreus, K. & Poels, G., 2011. A Goal-Oriented Requirements Engineering Method for Business Processes. In *Information Systems Evolution*. Springer Berlin Heidelberg, pp. 29–43.
- Delfmann, P., 2009. Unified Enterprise Knowledge Representation with Conceptual Models - Capturing Corporate Language in Naming Conventions. In *ICIS 2009 Proceedings*. pp. 1–16.
- Dijkman, R.M.R.M., Dumas, M. & Ouyang, C., 2007. Formal semantics and automated analysis of BPMN process models. *Information and Software Technology*, pp.1281–1294.
- Doerr, M., Hunter, J. & Lagoze, C., 2003. Towards a Core Ontology for Information Integration. *Journal of Digital Information*, 4(1).
- Dumas, M. et al., 2013. *Fundamentals of Business Process Management*, Springer Berlin / Heidelberg.
- Dunnette, M., Campbell, J. & Jaastad, K., 1963. The effect of group participation on brainstorming effectiveness for two industrial samples. *Journal of Applied Psychology*, 47(1), pp.30–37.
- Endert, H. et al., 2007. Mapping BPMN to Agents: An analysis. In M. Baldoni, C. Baroglio, & V. Mascardi, eds. *International Workshop MALLOW-AWESOME'007*. pp. 43–58.
- Euzenat, J. & Shvaiko, P., 2013. *Ontology matching*, Springer Berlin / Heidelberg.
- Evermann, J. & Wand, Y., 2005a. Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering*, 10(2), pp.146–160.
- Evermann, J. & Wand, Y., 2005b. Toward formalizing domain modeling semantics in language syntax. *Ieee Transactions on Software Engineering*, 31(1), pp.21–37.
- Fensel, D. et al., 2000. OIL in a Nutshell. In *Knowledge engineering and knowledge management methods, models, and tools*. Springer Berlin Heidelberg, pp. 1–16.
- Fernández-López, M., Gómez-Pérez, A. & Juristo, N., 1997. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Spring Symposium on Ontological Engineering of AAAI*. pp. 33–40.
- Fettke, P. & Loos, P., 2003. Ontological Evaluation of Reference Models using the Bunge-Wand-Weber Model. In *Proceedings of the 9th Americas Conference on Information Systems*.
- Fill, H.-G., 2012. An Approach for Analyzing the Effects of Risks on Business Processes Using Semantic Annotations. *ECIS 2012 Proceedings*.

- Fill, H.-G., 2011a. On the Conceptualization of a Modeling Language for Semantic Model Annotations. In C. Salinesi & O. Pastor, eds. *Advanced Information Systems Engineering Workshops*. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 134–148.
- Fill, H.-G., 2011b. Using Semantically Annotated Models for Supporting Business Process Benchmarking. In J. Grabis & M. Kirikova, eds. *Perspectives in Business Informatics Research*. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 29–43.
- Fonseca, F. & Martin, J., 2007. Learning The Differences Between Ontologies and Conceptual Schemas Through Ontology-Driven Information Systems. *Journal of the Association for Information Systems*, 8(2), pp.129–142.
- Fox, M., 1992. The TOVE Project Towards a Common-Sense Model of the Enterprise. In F. Belli & F. Radermacher, eds. *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Springer Berlin Heidelberg, pp. 25–34.
- Francescomarino, C. Di et al., 2011. A framework for the collaborative specification of semantically annotated business processes. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4), pp.261–295.
- Francescomarino, C. Di & Tonella, P., 2009. Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. In *Enterprise, Business Process, and Information Systems Modelling*. pp. 211–223.
- Di Francescomarino, C. & Tonella, P., 2010. Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. *International Journal of Information System Modeling and Design*, 1(2), pp.59–84.
- Franch, X., 2010. Fostering the Adoption of i * by Practitioners : Some Challenges and Research Directions. In S. Nurcan et al., eds. *Intentional Perspectives on Information Systems Engineering*. Springer Berlin Heidelberg, pp. 177–193.
- Franch, X. et al., 2011. Ontological Analysis of Means-End Links. In *CEUR Workshop Proceedings*.
- Gailly, F., 2016. Recommendation-based Conceptual Modeling and Ontology Evolution (CMOE+) Java tool.
- Gailly, F., Geerts, G. & Poels, G., 2009. Ontological Reengineering of the REA-EO Using UFO. In *International OOPSLA Workshop on Ontology-Driven Software Engineering*.
- Gangemi, A. et al., 2002. Sweetening ontologies with DOLCE. *Knowledge Engineering and Knowledge Management, Proceedings*, 2473, pp.166–181.
- Geerts, G.L. & McCarthy, W.E., 1999. An accounting object infrastructure for knowledge- based enterprise models. *Intelligent Systems and their Applications, IEEE*, 14(4), pp.89–94.
- Gehlert, A. & Esswein, W., 2007. Toward a formal research framework for ontological analyses. *Advanced Engineering Informatics*, 21(2), pp.119–131.
- Gemino, A. & Wand, Y., 2004. A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, 9(4), pp.248–260.
- Glimm, B. et al., 2014. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3), pp.245–269.
- Gómez-Pérez, a. & Rojas-Amaya, M., 1999. Ontological reengineering for reuse. In D. Fensel & R. Studer, eds. *11th European Workshop in Knowledge Acquisition, Modeling and Management*. Dagstuhl Castle, Germany: Springer Verlag, pp. 139–156.
- Gomez - Perez, A., Fernandez - Lopez, M. & Corch, O., 2003. *Ontological engineering* X. Wu & L. Jain, eds., Springer.
- Gordijn, J. & Akkermans, H., 2001. Designing and Evaluating E-Business Models. *IEEE Intelligent Systems*, 16(4), pp.11–17.
- Gordijn, J. & Akkermans, J.M., 2001. E3-value: Design and Evaluation of e-Business Models. *IEEE Intelligent Systems*, 16(4), pp.11–17.
- Grau, B.C. et al., 2008. OWL 2: The next step for OWL. *Web Semantics*, 6(4), pp.309–322.
- Gruber, T.R., 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Creation Diffusion Utilization*, 5(April), pp.199–220.

- Guarino, N., 1998. Formal Ontology and Information Systems. , (June), pp.3–15.
- Guarino, N. & Welty, C., 2002a. Evaluating ontological decisions with ONTOCLEAN. *Communications of the ACM - Ontology: Different Ways of Representing the Same Concept*, 45(2), pp.61–65.
- Guarino, N. & Welty, C., 2002b. Evaluating ontological decisions with ONTOCLEAN. , 45(2), pp.61–65.
- Guizzardi, G., 2012. Ontological Foundations for Conceptual Modeling with Applications. In J. Ralyté et al., eds. *Advanced Information Systems Engineering*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 695–696.
- Guizzardi, G., 2005. *Ontological foundations for structural conceptual models*.
- Guizzardi, G., 2013. Ontology-Based Evaluation and Design of Visual Conceptual Modeling Languages I. Reinhartz-Berger et al., eds. *Domain Engineering. Product Lines, Languages and Conceptual Models*, (i), p.345.
- Guizzardi, G. et al., 2015. Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology*, 10(3–4), pp.259–271.
- 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*. Springer Berlin / Heidelberg, pp. 100–115.
- Guizzardi, G. & Wagner, G., 2010a. Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In R. Poli et al., eds. *Theory and Applications of Ontology*. Springer Berlin, pp. 175–196.
- Guizzardi, G. & Wagner, G., 2010b. Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In R. Poli et al., eds. *Theory and Applications of Ontology*. Springer Berlin, pp. 175–196.
- Guizzardi, G. & Wagner, G., 2010c. Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In R. Poli et al., eds. *Theory and Application of Ontologies*. Berlin: Springer, pp. 175–196.
- Guizzardi, G. & Wagner, G., 2008. What ' s in a Relationship : An Ontological Analysis. In *Lecture Notes in Computer Science*. pp. 83–97.
- Guizzardi, R. et al., 2008. Ontological Foundations for Agent-Oriented Organizational Modeling. In *3rd International IStar Workshop*. pp. 37–41.
- Guizzardi, R. & Reis, A., 2015. A method to Align Goals and Business Processes. In *International Conference on Conceptual Modeling*. pp. 79–93.
- Gupta, U.G. & Clarke, R.E., 1996. Theory and applications of the Delphi technique: A bibliography (1975–1994). *Technological Forecasting and Social Change*, 53(2), pp.185–211.
- Hahn, A., 2005. Integration verteilter Produktmodelle durch Semantic-Web-Technologien. *Wirtschaftsinformatik*, 47(4), pp.278–284.
- Hammer, M., 1990. Reengineering work: Dont't automate, obligate. *Harvard Business Review*, 68(4), pp.104–112.
- Harzallah, M., Berio, G. & Opdahl, A.L., 2012. New perspectives in ontological analysis: Guidelines and rules for incorporating modelling languages into UEML. *Information Systems*, 37(5), pp.484–507.
- van Heijst, G., Schreiber, A. & Wielinga, B., 1997. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 45, pp.183–292.
- Hepp, M. et al., 2005. Semantic business process management: a vision towards using semantic Web services for business process management. In *IEEE International Conference on Business Process Engineering*. IEEE Computer Society, pp. 535–540.
- Hepp, M. et al., 2005. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*. Beijing, CHINA: IEEE, pp. 535–540.
- Hepp, M., Bachlechner, D. & Siorpaes, K., 2006. OntoWiki: Community-driven Ontology Engineering and Ontology Usage Based on Wikis. In *Proceedings of the 2006 international symposium on Wikis*. pp. 143–144.

- Hepp, M. & Roman, D., 2007. An Ontology Framework for Semantic Business Process Management. *Wirtschaftsinformatik*, pp.1–18.
- Hevner, A.R. et al., 2004. Design Science in Information Systems Research. *MIS Quarterly*, 28(1), pp.75–105.
- Hofferer, P., 2007. Achieving Business Process Model Interoperability Using Metamodels and Ontologies. In *European Conference on Information Systems*. pp. 1620–1631.
- Holsapple, C.W. & Joshi, K.D., 2002. A collaborative approach to ontology design. *Communications of the ACM*, 45(2), pp.42–47.
- Horrocks, I., Patel - Schneider, P. & van Harmelen, F., 2003. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1), pp.7–26.
- IEEE, 1997. *IEEE Standard for Developing Software Life Cycle Processes*, New York: IEEE.
- IEEE Standard Upper Ontology Working Group, Suggested Upper Merged Ontology, 2006
- Jonkers, H. et al., 2004. Concepts for modelling Enterprise Architectures. *International Journal of Cooperative Information Systems*, 13(Nadler 1992), pp.257–287.
- Jureta, I., Mylopoulos, J. & Faulkner, S., 2009. A core ontology for requirements. *Applied Ontology*, 4(3), pp.169–244.
- Kaneiwa, K., Iwazume, M. & Fukuda, K., 2007. An Upper Ontology for Event Classifications and Relations. In *20th Australian Joint Conference on Advances in Artificial Intelligence*. pp. 394–403.
- Katsumi, M. & Gruninger, M., 2016. What is ontology reuse? In R. Ferrario & W. Kuhn, eds. *Formal Ontology in Information Systems*. IOS Press, pp. 9–22.
- Kesh, S., 1995. Evaluating the quality of entity relationship models. *Information and Software Technology*, 37(12), pp.681–689.
- Kietz, J.-U., Meadche, A. & Volz, R., 2000. A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In N. Aussenac-Gilles, B. Biebow, & S. Szulman, eds. *Workshop on Ontologies and Texts EKAW'00*. Juan-les-Pins, France: CEUR Workshop Proceedings, p. 51:4.1-4.14.
- Koliadis, G. et al., 2006. Combining i* and BPMN for Business Process Model Lifecycle Management. In *BPM 2006 Workshops*. pp. 416–427.
- Kotis, K. & Vouros, G.A., 2006. Human-centered ontology engineering: The HCOME methodology. *Knowledge and Information Systems*, 10(1), pp.109–131.
- Kueng, P., Bichler, P. & Kawalek, P., 1996. How to Compose an Object-oriented Business Process Model? *Method Engineering: Principles of method construction and tool support*, pp.94–110.
- Van Lamsweerde, A., 2001. Goal-oriented requirements engineering: a guided tour. *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pp.249–262.
- van Lamsweerde, A., Darimont, R. & Letier, E., 1998. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11), pp.908–925.
- Lankhorst, M., 2005. *Enterprise architecture at work : modelling, communication, and analysis*, Berlin ; New York: Springer.
- Lapouchnian, A., Yu, Y. & Mylopoulos, J., 2007. Requirements-Driven Design and Configuration Management of Business Processes. *Business Process Management*, pp.246–261.
- Leutgeb, A. et al., 2007. Adaptive Processes in E-Government - A Field Report about Semantic-Based Approaches from the EU-Project FIT. *ICEIS 2007 - Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume EIS, Funchal, Madeira, Portugal, June 12-16, 2007*, pp.264–269.
- Levitin, A. & Redman, T., 1995. QUALITY DIMENSIONS OF A CONCEPTUAL VIEW. *Science*, 31(1), pp.81–88.
- Lin, F.-R., Yang, M.-C. & Yu-Hua, P., 2002. A generic structure for business process modeling. *Business Process Management Journal*, 8(1), pp.19–41.
- Lin, Y. et al., 2006. Semantic annotation framework to manage semantic heterogeneity of process models. In E. Dubois & K. Pohl, eds. *18th International Conference on Advanced Information Systems Engineering*. Luxembourg: Springer-Verlag, pp. 433–446.

- Lin, Y. & Strasunskas, D., 2005. Ontology-based semantic annotation of process templates for reuse. In *CEUR Workshop Proceedings*. pp. 207–218.
- Lindland, O.I., Sindre, G. & Solvberg, A., 1994. Understanding quality in conceptual modeling. *IEEE Software*, 11(2), pp.42–49.
- Luczak-Rösch, M. et al., 2010. SVoNt-Version Control of OWL Ontologies on the Concept Level. In *Proceedings of the 5th International Workshop on Applications of Semantic Technologies*. pp. 79–84.
- Maedche, A. & Staab, S., 2000. Semi - automatic engineering of ontologies from text. In S. Chang & W. Obozinski, eds. *12th International Conference on Software Engineering and Knowledge Engineering SEKE'00*. Chicago.
- Maes, A. & Poels, G., 2007. Evaluating quality of conceptual modelling scripts based on user perceptions. *Data & Knowledge Engineering*, 63(3), pp.701–724.
- Di Martino, B. et al., 2016. A methodology and implementing tool for semantic business process annotation. In *International Workshop on Business Process Modelling, Development and Support*. Springer International Publishing, pp. 80–94.
- Matulevičius, R., Heymans, P. & Opdahl, A.L., 2007. Ontological analysis of KAOS using separation of reference. In K. Siau, ed. *Contemporary Issues in Data Base Design and Information Systems Development*. pp. 37–54.
- Matulevicius, R., Heymans, P. & Sindre, G., 2015. Comparing Goal-Modelling Tools With the Re-Tool Evaluation Approach. *Information Technology and Control*, 35(3), pp.276–286.
- Mehmood, K., Si-Said Cherfi, S. & Comyn-Wattiau, I., 2009. Data Quality Through Conceptual Model Quality - Reconciling Researchers and Practitioners Through a Customizable Quality Model. In *International Conference on Information Quality (ICIQ)*. pp. 61–74.
- Mens, T., Van Der Straeten, R. & Simmonds, J., 2005. A Framework for Managing Consistency of Evolving UML Models. In *Software Evolution with UML and XML*. pp. 1–30.
- Miller, G. a., 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11), pp.39–41.
- Missikoff, M., Smith, F. & Taglino, F., 2015. Ontology building and maintenance in collaborative virtual environment. *Concurrency Computation Practice and Experience*, 27(11), pp.2796–2817.
- Moody, D., 2003. The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods. In *11th European Conference on Information Systems, ECIS 2003*. Naples, Italie, pp. 1327–1326.
- Moody, D., Shanks, G. & Darke, P., 1998. Improving the Quality of Entity Relationship Models — Experience in Research and Practice. *Quality*, pp.255–276.
- Moody, D.L., 2005. Theoretical and practical issues in evaluating the quality of conceptual models : current state and future directions. *Data & Knowledge Engineering*, 55(3), pp.243–276.
- Moody, D.L., Heymans, P. & Matulevičius, R., 2010. Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. *Requirements Engineering*, 15(2), pp.141–175.
- de Moor, A., De Leenheer, P. & Meersman, R., 2006. DOGMA-MESS: A Meaning Evolution Support System for Interorganizational Ontology Engineering. In H. Schärfe, P. Hitzler, & P. Øhrstrøm, eds. *Conceptual Structures: Inspiration and Application SE - 14*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 189–202.
- Moor, A. De, Leenheer, P. De & Meersman, R., 2006. DOGMA-MESS : A Meaning Evolution Support System for Interorganizational Ontology Engineering. In H. Scharfe, P. Hitzler, & P. Ohrstrom, eds. *Conceptual Structure: Inspiration and Application*. Springer, pp. 189–202.
- Mostow, J., 1985. Toward Better Models Of The Design Process. *AI Magazine*, 6(1), pp.44–57.
- zur Muehlen, M., Indulska, M. & Kamp, G., 2007. Business Process and Business Rule Modeling : A Representational Analysis. In *EDOC Conference Workshop*. pp. 189–196.
- Mylopoulos, J., 1992. Conceptual Modelling and Telos. In P. Loucopoulos & R. Zicari, eds. *Conceptual modeling, databases, and CASE: an integrated view on information system development*. John Wiley & Sons, pp. 49–68.

- Nagel, B. et al., 2013. Ensuring consistency among business goals and business process models. *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, pp.17–26.
- Nardi, J.C. et al., 2013. Towards a commitment-based reference ontology for services. In *Enterprise Distributed Object Computing Conference*. IEEE, pp. 175–184.
- Neches, R. et al., 1991. Enabling Technology for Knowledge Sharing. , 12(3).
- Nelson, H.J. et al., 2011. A conceptual modeling quality framework. *Software Quality Journal*, 20(1), pp.201–228.
- Niles, I. & Pease, A., 2013. Linking Lexicons and Ontologies : Mapping WordNet to the Suggested Upper Merged Ontology. In *International Conference on Information and Knowledge Engineering*. pp. 412–416.
- Niles, I. & Pease, A., 2001. Towards a Standard Upper Ontology. In *The 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*. pp. 2–9.
- Noy, N. & Musen, M., 2004. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems*, 19(4), pp.6–13.
- Noy, N.F. & Musen, M. a, 1999. SMART : Automated Support for Ontology Merging and Alignment. In B. Gaines, B. Kremer, & M. Musen, eds. *12th Banff Workshop on Knowledge Acquisition, Modelling and Management*. Banff, Canada, pp. 4-7-20.
- Object Management Group (OMG), 2008. *Business Process Model and Notation , V1.1*, Available at: <http://www.omg.org/spec/BPMN/1.1/PDF>.
- OMG, 2011. *Business Process Model and Notation (BPMN) Version 2.0*, Available at: <http://books.google.com/books?id=GjmLqXNYFS4C&pgis=1>.
- OMG, 2006a. Business Process Modeling Notation Specification. , (February), p.308. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.4777&rep=rep1&type=pdf>.
- OMG, 2006b. Business Process Modeling Notation Specification (dtc/06-01-01). , (dtc/06-01-01).
- OMG, 2007. OMG Unified Modeling Language (OMG UML), superstructure, V2.1.2. , (November). Available at: <http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF>.
- Opdahl, A.L. et al., 2012. An ontology for enterprise and information systems modelling. *Applied Ontology*, 7(1), pp.49–92.
- Opdahl, A.L. & Henderson-Sellers, B., 2002. Ontological Evaluation of the UML Using the Bunge–Wand–Weber Model. *Software and Systems Modeling*, 1(1), pp.43–67.
- Osterwalder, A. & Pigneur, Y., 2002. An e-Business Model Ontology for Modeling e-Business. In *15th Electronic Commerce Conference e-reality: Constructing the e-Economy*. Bled, Slovenia.
- Pedrinaci, C., Domingue, J. & de Medeiros, A.K., 2008. A core ontology for business process analyses. In S. Bechhofer et al., eds. *ESWC'08 Proceedings of the 5th European semantic web conference on The semantic web: research and applications*. Springer Berlin, Heidelberg, pp. 49–64.
- Pfeiffer, D., 2007. Constructing comparable conceptual models with domain specific languages. *Proceedings of the 15th European Conference on Information Systems, ECIS 2007*, pp.876–888.
- Pichler, P. et al., 2011. Imperative versus declarative process modeling languages: An empirical investigation. In *International Conference on Business Process Management*. Springer Berlin Heidelberg, pp. 383–394.
- Pinggera, J., Zugal, S. & Weber, B., 2010. Investigating the process of process modeling with cheetah experimental platform- Tool paper. In *CEUR Workshop Proceedings*. pp. 13–18.
- Pittke, F., Leopold, H. & Mendling, J., 2013. Spotting Terminology Deficiencies in Process Model Repositories. In S. Nurcan et al., eds. *Enterprise, Business Process and Information Systems Modelling*. Valencia, pp. 292–307.
- Rolón, E. et al., 2006. Applying Software Metrics to evaluate Business Process Models. *CLEI-Electronic Journal*, 9(1), Paper 5.
- Rosemann, M., 2006. Potential pitfalls of process modeling: part A. *Business Process Management Journal*, 12(2), pp.249–254.

- Rosemann, M. et al., 2015. The Six Core Elements of Business Process Management. In J. vom Brocke & M. Rosemann, eds. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 105–122.
- Rosemann, M. & Green, P., 2002. Developing a meta model for the Bunge–Wand–Weber ontological constructs. *Information Systems*, 27(2), pp.75–91.
- Rospocher, M., Ghidini, C. & Serafini, L., 2014. An ontology for the business process modelling notation. In *Frontiers in Artificial Intelligence and Applications*. pp. 133–146.
- Ruy, F.B. et al., 2015. Ontology Engineering by Combining Ontology Patterns. In pp. 173–186.
- Sánchez-González, L. et al., 2013. Toward a Quality Framework for Business Process Models. *International Journal of Cooperative Information Systems*, 22(1).
- Santos, E. et al., 2010. A Goal-Oriented Approach for Variability in BPMN. In *Workshop em Engenharia de Requisitos*. pp. 17–28.
- Santos, P., Almeida, J. & Guizzardi, G., 2013. An ontology-based analysis and semantics for organizational structure modeling in the ARIS method. *Information Systems*, 38(5), pp.690–708.
- Schaffert, S., 2006. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*. IEEE, pp. 388–396.
- Seidewitz, E., 2003. What models mean. In *IEEE Software*. pp. 26–32.
- Shanks, G. & Darke, P., 1997. Quality in conceptual modelling: linking theory and practice. In *PACIS 1997 Proceedings*. p. 76.
- Shibaoka, M., Kaiya, H. & Saeki, M., 2007. GOORE : Goal-Oriented and Ontology Driven Requirements Elicitation Method. *Advances in Conceptual Modeling – Foundations and Applications*, 4802, pp.225–234.
- Si-Said Cherfi, S., Ayad, S. & Comyn-Wattiau, I., 2013. Improving Business Process Model Quality Using Domain Ontologies. *Journal on Data Semantics*, 2(2–3), pp.75–87.
- Silver, B., 2009. *BPMN Method and Style*, Cody - Cassidy Press.
- Simperl, E. & Luczak-Rösch, M., 2014. Collaborative ontology engineering: a survey. *Knowledge Engineering Review*, 29(1), pp.101–131.
- Simperl, E.P.B. & Tempich, C., 2006. Ontology engineering: a reality check. In *OTM confederated International Conferences "On the move to meaningful Internet systems."* Springer Berlin Heidelberg, pp. 836–854.
- Sonnenberg, C. et al., 2011. The REA-DSL : A Domain Specific Modeling Language for Business Models. In H. Mouratidis & C. Rolland, eds. *International Conference on Advanced Information Systems Engineering*. London: Springer Berlin Heidelberg, pp. 252–266.
- Sousa, H.P. et al., 2014. Modeling Organizational Alignment. In E. Yu et al., eds. *Conceptual Modeling: 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*. Cham: Springer International Publishing, pp. 407–414.
- Sowa, JA and Zachman, J.A., 1992. the framework for information systems architecture. *IBM Systems Journal*, 31(3), pp.590–616.
- Studer, R., Benjamins, V.R. & Fensel, D., 1998. Knowledge Engineering : Principles and Methods. *Data and Knowledge Engineering*, 25(1), pp.161–197.
- Suárez-Figueroa, M.C. et al., 2012. *Ontology Engineering in an Networked World*, Springer Berlin / Heidelberg.
- The Open Group, 2013. ArchiMate.
<https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12480>.
- The Open Group, The Open Group Architecture Framework (TOGAF).
<https://www2.opengroup.org/ogsys/catalog/g116>.
- Thomas, O. & Fellmann M.A., M., 2009. Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. *Business & Information Systems Engineering*, 1, pp.438–451.

- 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. *Business & Information Systems Engineering*, 1(6), pp.438–451.
- Tudorache, T. et al., 2008. Supporting Collaborative Ontology Development in Protege. In A. Sneth et al., eds. *Proceedings of the 7th International Conference on the Semantic Web*. Springer, pp. 17–32.
- Uschold, M. et al., 1996. The Enterprise Ontology. *The Knowledge Engineering Review*, 13(1), pp.31–89.
- Uschold, M. et al., 1998. The Enterprise Ontology. *The Knowledge Engineering Review: Special Issue on Putting Ontologies to Use*, 13(1), pp.31–89.
- Uschold, M. & Jasper, R., 1999. A Framework for Understanding and Classifying Ontology Applications. In *IJCAI Workshop on Ontologies and Problem-Solving Methods*.
- Vazquez, B., Martinez, A., et al., 2013. Enriching Organizational Models through Semantic Annotation. *Journal of Materials Processing Tech.*, 7, pp.297–304.
- Vazquez, B., Martínez, A., et al., 2013. Towards supporting business services discovery through the integration of organizational models with ontologies. *CibSE 2013*, pp.215–227.
- Vrandecic, D. et al., 2005. The DILIGENT knowledge processes. *Journal of Knowledge Management*, 9(5), pp.85–96.
- Wand, Y. & Weber, R., 1988. An ontological analyses of some fundimantal information systems concepts. In *Proceedings on the 9th International Conference on Information Systems*. pp. 213–225.
- Wand, Y. & Weber, R., 1990. An ontological model of an information system. *IEEE Transactions on Software Engineering*, 16(11), pp.1282–1292.
- Wand, Y. & Weber, R., 2002. Research Commentary : Information Systems and Conceptual Modeling — A Research Agenda. *Information Systems Research*, (4), pp.363–376.
- Wetzstein, B. et al., 2007. Semantic business process management: a lifecycle based requirements analysis. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management*. pp. 1–11.
- White, S.A., 2004. *Introduction to BPMN*,
- Wieringa, R.J. & Heerkens, J.M.G., 2006. The methodological soundness of requirements engineering papers: a conceptual framework and two case studies. *Requirements Engineering*, 11(4), pp.295–307.
- Winkler, W., 1990. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage.
- Winter, S., 2001. Ontology: Buzzword or Paradigm Shift in GI Science? *International Journal of Geographical Information Science*, 7(15), pp.587–590.
- Yu, E. et al., 2013. Practical applications of i* in industry: The state of the art. In *IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*. pp. 366–367.
- Yu, E. et al., 2011. *Social Modelling for Requirements Engineering*, Cambridge: MIT Press.
- Yu, E., Strohmaier, M. & Deng, X., 2006. Exploring Intentional Modeling and Analysis for Enterprise Architecture. In *10th IEEE International Enterprise Distributed Object Computing Conferenace Workshops*. IEEE, p. 32.
- Yu, E.S., 2009. Social Modeling and i *. In *Conceptual Modelling: Foundations and Applications*. pp. 99–121.
- Yu, E.S.K., 1997. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*. IEEE Comput. Soc. Press, pp. 226–235.
- Zablith, F. et al., 2015. Ontology evolution: a process-centric survey. *The Knowledge Engineering Review*, 30(1), pp.45–75.
- Zave, P. & Jackson, M., 1997. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1), pp.1–30.

A


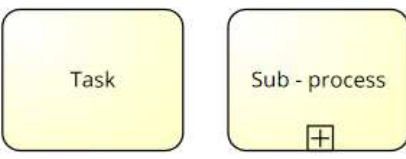
BPMN Constructs

A BPMN diagram constitutes of four categories of modeling elements: Flow objects, Connecting objects, Swimlanes, and Artifacts.

Flow objects

This group includes the core elements to BPMN diagrams. Every element and its description is presented in Table 1 below.

Table 1: BPMN flow objects

Event	Event is something which happens during the process. It typically has a trigger or an impact. An event can be start, intermediate, and end event based on its location in the process (either beginning, middle, or end respectively).	 Start event Intermediate event End event
Activity	Activity is a generic term applicable to the work being performed within an organization. An atomic activities is named "task", while a compound activity which can be further decomposed is called "sub – process".	 Task Sub - process

Gateway	<p>Gateway is used to control the flow of the process. One gateway can have multiple inputs and / or multiple outputs. The main gateway types are:</p> <ul style="list-style-type: none"> • Exclusive gateway which allows alternative paths, and can be compared to a divergence point on the road • Inclusive gateway can create allows parallel flow in addition to alternative paths • Parallel gateway is used to synchronize parallel flow • Event – based gateway where a path followed after the gateway is determined by occurrence of a predefined event 	<p>Exclusive gateway Inclusive gateway</p> <p>Parallel gateway Event - based gateway</p>
---------	--	--

Connecting objects

The previously described flow objects are connected together by connecting objects. Connecting objects are divided into three categories (Table 2) based on objects they are connecting.


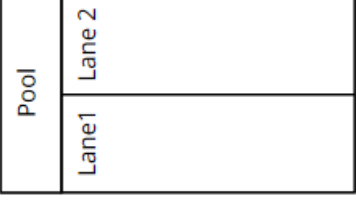
Table 2: BPMN connecting objects

Sequence flow	Sequence flow indicates the order in which activities and events are occurring.	
Message flow	Message flow represents the exchange of messages among two participants of the same process. A participant can be a business entity, a person, artificial agent, etc.	
Association	Association is solely used to associate data objects (such as documents) with corresponding flow objects .	

Swimlanes

Swimlanes (displayed in Table 3) are used to group flow objects into visual categories. Very often swimlanes represent different participants in the process.




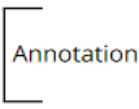
Table 3: Swimlanes

Pool	Pool represents process participant, for example, business entity	
Lane	Lane is a sub – partition of the pool. It is used to represent different roles and divisions within a business entity, or to create sub – group of activities within a process.	

Artifacts

Artifacts were included as a part of BPMN to allow the modeler more flexibility. It is possible to add as many artifacts as needed to the process diagram. Artifacts allowed within BPMN notation are presented in Table 4 below.

Table 4: BPMN artifacts

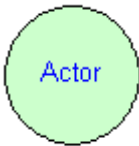

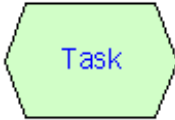
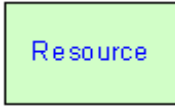

Data object	This artifact allows the modeler to represent all the data (such as reports) consumed and generated during the course of the process.	
Data store	Using a data store the modeler can express which activities access or update stored information.	
Group	Grouping is used for documentation and analyses purpose.	
Annotation	Annotation is used to allow the modeler to supply additional text for the reader.	

B

I* Constructs

Table 5 presents only the basic set of i* constructs which is sufficient to follow this doctoral thesis.

Table 5: I* basic constructs

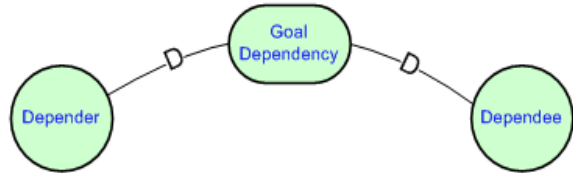
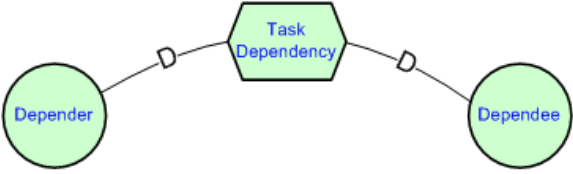
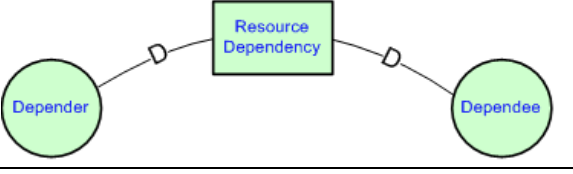
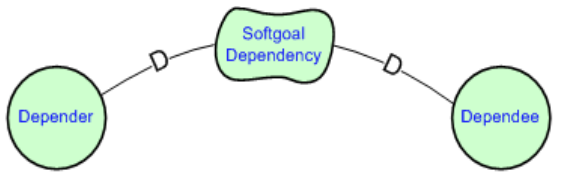
Actor	Actor is an entity which carries out actions to achieve goals. Actor is a generic term which can represent a human, an artificial agent, or an organization.	
Goal	Describes the state of affairs as desired by a particular actor. The goal itself does not prescribe a specific path for achieving it.	
Task	A task to be accomplished by a particular actor.	
Resource	A provision of some entity physical or informational, which is required for achieving a goal or executing a task.	
Softgoal	Softgoal is similar to a Goal construct. However, the satisfaction of a softgoal is not a clear – cut. It is evaluated	

	from actor's point of view.	
--	-----------------------------	--

SD model

In this type of i* model one actor (a depender), depends on another actor (a dependee) to acquire a dependum which can be goal, task, resource, or softgoal. An overview of possible dependencies can be found in Table 6.

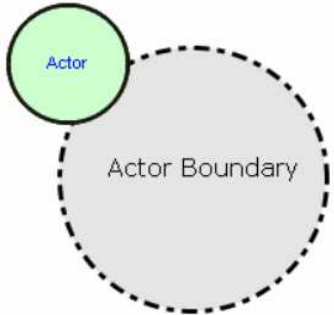
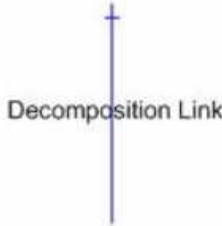
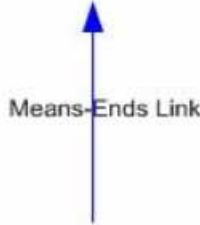
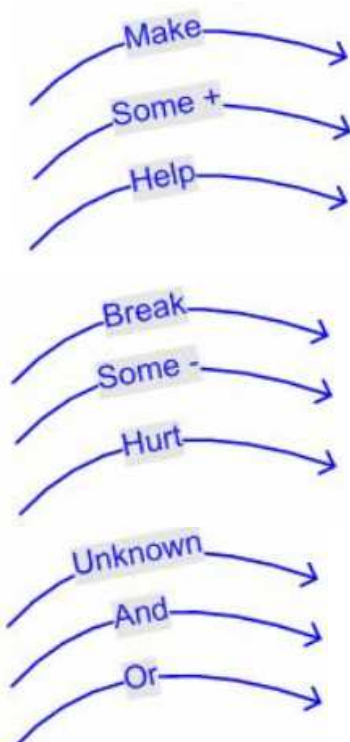
Table 6: Dependencies represented in SD

Goal dependency	In this type of dependency the depender depends on the dependee to achieve a particular state of affairs.	
Task dependency	Here the dependency is required to carry out an activity.	
Resource dependency	In resource dependency one actor depends on the other for acquisition of an entity.	
Softgoal dependency	In this dependency the dependee is expected to perform a task that meets a softgoal put forward by the depender.	

SR model

Strategic rationale model offers a look “inside” actors, thereby enabling the modeler to model internal intentional relationships. This model allows representing tasks that need to be completed in order to achieve a particular goal. It enables the modeler to show alternative ways for achieving a goal, and finally, it depicts how various elements within the model contribute to a particular softgoals. This contribution can be positive or negative. The notation is presented in Table 7 below.

Table 7: SR model

Actor boundary	A boundary surrounding all the elements explicitly desired by that particular actor.	
Decomposition link	This link allows decomposition of a task into sub – tasks, sub – goals, resources, or softgoals. One task can be decomposed into one or more element.	
Means - ends link	This link indicates alternative paths which can be undertaken to achieve a goal. An end is a goal, and a mean is usually a task.	
Contribution link	Contribution link specifies an extent to which other i* elements are contributing to the fulfillment of a softgoal. This contribution can be positive (such as “some” and “make” links), or it can be negative as it is the case in “hurt” and “break” link.	

Unified Foundational Ontology (UFO)

To exemplify the recommendation-based business process modeling, 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 Universal* 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).

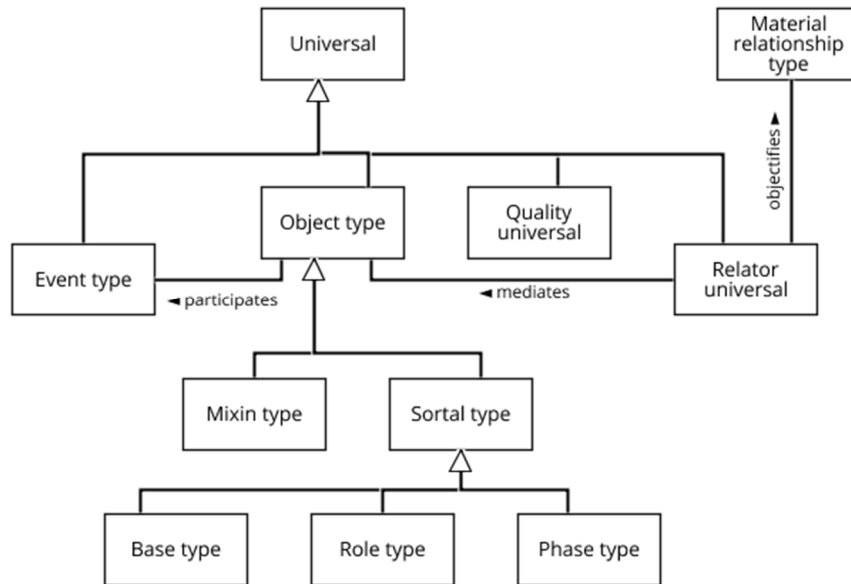


Figure 1: Fragment UFO

D

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	BaseType

		Report	
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

E

Loan Application 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.

F

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).

G

Post-survey

Questions of the post-survey classified according to the dependent variables to be measured:

PEOU1: I often made errors while modeling 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

H

Reference Model

