Ontology Based Semantic Annotation of Business Processes with Semi-Automated Suggestions and Processing of Semantic Queries in Business Process Models

Ioanna S. Ramoutsaki

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes Campus, GR-70013 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Dimitris Plexousakis*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.
The work has been supported by the Foundation for Research and Technology _ Hellas (FORTH), Institute of Computer Science (ICS).

UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

Ontology Based Semantic Annotation of Business Processes with Semi-Automated Suggestions and Processing of Semantic Queries in Business Process Models

Thesis submitted by
Ioanna S. Ramoutsaki
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: ________________________
Ioanna S. Ramoutsaki

Committee approvals: ________________________
Dimitris Plexousakis
Professor, Thesis Supervisor

____
Antonis Savidis
Professor, Committee Member

____
Kostas Magoutis
Assistant Professor, Committee Member

Departmental approval: ________________________
Antonis Argyros
Professor, Director of Graduate Studies

Heraklion, December 2016

Abstract

Despite increasing software support for Business Process Modeling, there are still misunderstandings, mainly because of terminology mismatches in business process models. Enriching business process models with semantic annotations derived from concepts of a domain ontology aims at overcoming this weakness of Business Process Modeling.

In this work, we present an approach for adding semantic annotations to business process elements, especially in activities (tasks and sub-processes) of business processes, as they are the main elements of a business process model, using an ontology-based data matching strategy. Semantic annotations are automatically suggested to the business designer, based on a composed measure of similarity between ontology concepts and the activity label of the process to be annotated. The combined similarity measure is an aggregation of the degrees returned from three similarity measures, aka string, linguistic and syntactic. Each one of the aforementioned is a priori assigned a specifically weight by the system. Finally, filtering techniques are used to display options with high relevance only.

By adding semantic information to process elements the precision of business process models increases, making them more intelligible to people and machine-readable, enabling automated reasoning services, such as querying the process space. For this purpose, we propose an automated approach for querying a business process model repository for semantically relevant elements and models. A basic BPMN ontology in OWL format has been developed for the needs of querying.

Our approach has been implemented in SeMFIS tool, which has been extended to represent the new functionalities of automated semantic annotations of business process models. The BPMN language has been used for the representation of business process models, the OWL format for ontologies and the SPARQL for queries.

Σημασιολογικός υπομνηματισμός βασιζόμενος σε οντολογία επιχειρησιακών διεργασιών με ημι-αυτοματοποιημένες προτάσεις και επεξεργασία σημασιολογικών επερωτήσεων σε μοντέλα επιχειρησιακών διεργασιών

Περίληψη

Παρόλο που έχουν αυξηθεί τα λογισμικά υποστήριξης για την μοντελοποίηση των επιχειρησιακών διεργασιών, υπάρχουν ακόμα προβλήματα κατανόησης τους, κυρίως εξαιτίας της αναντιστοιχίας ορολογιών στα μοντέλα επιχειρησιακών διεργασιών. Εμπλουτίζοντας τα μοντέλα των επιχειρησιακών διεργασιών με σημασιολογικούς υπομνηματισμούς που προέρχονται από έννοιες μιας ειδικής οντολογίας αποσκοπεί στην αντιμετώπιση αυτής της αδυναμίας της μοντελοποίησης επιχειρησιακών διεργασιών.

Σε αυτή την εργασία παρουσιάζουμε μία προσέγγιση για την προσθήκη σημασιολογικών υπομνηματισμών στα στοιχεία των επιχειρησιακών διεργασιών, και ειδικότερα στις "δραστηριότητες" των επιχειρησιακών διεργασιών (που αποτελούνται από τις "εργασίες" και τις "υπο-διεργασίες"), καθώς αυτές είναι τα κύρια στοιχεία ενός μοντέλου επιχειρησιακής διεργασίας, χρησιμοποιώντας μία στρατηγική αντιστοίχισης δεδομένων βασιζόμενη σε μία οντολογία. Οι σημασιολογικοί υπομνηματισμοί προτείνονται αυτόματα στον σχεδιαστή, ακολουθώντας μία μέθοδο που βασίζεται σε ένα σύνθετο μέτρο ομοιότητας μεταξύ των εννοιών της οντολογίας και του ονόματος της "δραστηριότητας" της διεργασίας που πρόκειται να υπομνηματιστεί. Το σύνθετο μέτρο ομοιότητας προκύπτει από την συνάθροιση τριών βαθμών που επιστρέφονται από τρία μέτρα ομοιότητας, των συμβολοσειρών, των γλωσσικών και των συντακτικών, στα οποία έχουν εκ των προτέρων αποδοθεί συντελεστές βαρύτητας από το σύστημα. Τέλος, στο τελικό μέτρο ομοιότητας χρησιμοποιούνται τεχνικές φιλτραρίσματος ώστε να εμφανιστούν στον χρήστη μόνο οι επιλογές που εμφανίζουν υψηλή συνάφεια με το όνομα της επιλεγμένης "δραστηριότητας".

Η προσθήκη σημασιολογικής πληροφορίας στα στοιχεία μία διεργασίας έχει ως αποτέλεσμα την αύξηση της ακρίβειας των μοντέλων επιχειρησιακών διεργασιών, καθιστώντας τα πιο κατανοητά στον άνθρωπο και αναγνώσιμα από την μηχανή, επιτρέποντας τη χρήση αυτοματοποιημένων υπηρεσιών "εξαγωγής συμπερασμάτων", όπως την χρήση επερωτήσεων στο χώρο των διεργασιών. Για το σκοπό αυτό, προτείνουμε μία προσέγγιση όπου αυτόματα θα γίνονται επερωτήσεις σ' ένα αποθετήριο μοντέλων επιχειρησιακών διεργασιών για την εύρεση σημασιολογικά

σχετικών στοιχείων και μοντέλων. Μία οντολογία, σε μορφή OWL, με τα βασικά στοιχεία της BPMN γλώσσας έχει αναπτυχθεί για τις ανάγκες της υπηρεσίας των επερωτήσεων.

Η προσέγγιση μας έχει υλοποιηθεί στο εργαλείο SeMFIS, το οποίο έχει επεκταθεί ώστε να συμπεριλάβει τις νέες λειτουργίες των αυτοματοποιημένων σημασιολογικών υπομνηματισμών στα μοντέλα επιχειρησιακών διεργασιών. Η BPMN γλώσσα έχει χρησιμοποιηθεί για την αναπαράσταση των μοντέλων επιχειρησιακών διεργασιών, η OWL μορφή για τις οντολογίες και η γλώσσα SPARQL για την υπηρεσία των επερωτήσεων.

Ευχαριστίες

Πρώτα από όλα θα ήθελα να ευχαριστήσω θερμά τον υπεύθυνο καθηγητή μου κ. Πλεξουσάκη όχι μόνο για την υποστήριξη, την καθοδήγηση, τις πολύτιμες συμβουλές και τον χρόνο που μου αφιέρωσε αλλά και γιατί μου έδειξε εμπιστοσύνη και δέχτηκε να γίνει ο επόπτης μου.

Θα ήθελα επίσης να ευχαριστήσω τον κ. Σαββίδη και τον κ. Μαγκούτη για την προθυμία τους να συμμετάσχουν στην τριμελή επιτροπή. Στη συνέχεια θα ήθελα να ευχαριστήσω την Δασκαλάκη Ευαγγελία από το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για το πολύτιμο υλικό που μου πρόσφερε αλλά και για τις συμβουλές της για τον τρόπο υλοποίησης της εργασίας μου.

Ένα μεγάλο ευχαριστώ στην οικογένεια μου που με την αγάπη, την κατανόηση και την υπομονή τους στηρίζουν κάθε νέα μου προσπάθεια. Θα κάνω ιδιαίτερη μνεία στον πατέρα μου, Στυλιανό, γιατί χωρίς το δικό του πιστεύω για τη δύναμη της γνώσης δεν θα είχα τα φτερά να συνεχίσω τις σπουδές μου.

Ένα ευχαριστώ οφείλω και στους φίλους μου για την στήριξή τους στις δύσκολες ώρες και την αμέριστη συμπαράστασή τους.

Το μεγαλύτερο όμως ευχαριστώ οφείλω στον σύντροφο μου, Βαγγέλη, ο οποίος ήταν καθ' όλη την διάρκεια των σπουδών μου δίπλα μου, να με στηρίζει, να υπομένει και να επιμένει, υπενθυμίζοντας μου πάντα ποιος είναι ο στόχος μου. Χωρίς αυτόν δεν θα τα είχα καταφέρει.

Στη μνήμη του μπαμπά μου Στυλιανού,

Στον σύντροφό μου Βαγγέλη,

Στον γιο μου Γιάννη

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Business Process Management (BPM) is a top-down methodology designed to organize, manage, analyze and reengineer the business processes running in enterprises [1]. The BPM is characterized by a lifecycle which begins with the process modeling phase. In this step the business designers create process models using a modeling tool. In the next step, namely implementation, the process model is translated by IT engineers to a workflow model, which runs on a process engine. The following step, the execution phase where the process engine executes the workflow model by delegating the process tasks to human workers or automated IT applications. Finally, monitoring tools are used by business analysts to measure the process performance which is called analysis phase [1].

Despite of increasing software support for BPM, there is still lack of automation in the BPM lifecycle. Specifically, there are general difficulties when it comes to bridge the gap between the business and IT views on the business processes. This happens because business analysts and IT experts do not speak the same language, do not share the same concepts of process or the same tools [1,2].

The Semantic Business Process Management (SBPM) is a new approach which attempts to close the Business - IT gap by integrating and utilizing semantic technologies in order to improve the level of automation in the modeling and management of business processes. Semantically annotated process models could enable support for the modeler in various tasks including reusing parts of process models when creating new one, detecting cross-process relations, providing a basis for knowledge transfer [1,2,3].

Within this context, this work focus on augmenting business processes with semi-automated suggestions of semantic annotations in the process modeling phase of BPM lifecycle using an ontology-based data matching strategy. The integration of semantic annotations in modeling tools will support the graphical modeling of business processes with concepts derived from domain ontologies which specify business process models more precisely, make them machine-readable, allow better understanding, documenting, querying and design choices that cannot be expressed in a purely syntactic way [4]. A lot of modeling languages and modeling tools have been implemented for the representation of processes. In this work, the Business Process

Modeling Notation language (BPMN), as a standard notation for capturing the business processes in the early phases of systems development, has been used to express the business process models and SeMFIS tool, as a flexible engineering platform for semantic annotations of conceptual models, has been extended to represent the new functionalities of automated semantic annotations of business process models (SBPM).

1.1 Motivation

Organizations have already invested heavily in business process management creating, most of the times, extensive business models [8]. However, enterprises face problems which cannot be solved with current business process management technologies because of the limited degree of mechanization in BPM, creating inertia in the necessary evolution and dynamics of business processes. In other words, both querying and manipulating the process space regularly requires human labor, leading many times to slow, costly and imperfect situations. It also does not provide a uniform representation of an organization's process space on a semantic level, which would be accessible to intelligent queries [6]. Moreover, when enterprises decide to interconnect business processes to perform common tasks, similar labeled process elements have to be identified to avoid misunderstandings. Although, using formal languages, such as BPMN, Petri Net, UML activity diagram for modeling business processes, purely syntactic composition problems of inter-organizational business environments may be solved, a missing semantic representation of process elements can prevent further interconnectivity and interoperability of business processes [7]. Therefore, Hepp et al. [9] suggest using the most significant results from the area of Semantic Web like ontologies, query languages, reasoners to provide additional support to the business process community. By describing business process models in a machine readable and interpretable format which enables semantic annotations and computer reasoning, the automation of process composition can be facilitated, helping organizations achieve the desired effectiveness, agility and ability to exploit opportunities. These also promise appropriate business process discovery, interoperability and interconnectivity which help to save costs and time when establishing inter-organizational business collaborations, as well as to accelerate finding appropriate composable business process models faster than manually discovering business process models [7].

1.2 Approach

During the graphical design, each business element is labeled according to arbitrarily information, resulting often in unclear labels, characterized by mismatching and overlapping terms and leads many times in loss of domain semantic knowledge. Labeling is not a rigorous task performed by the business designers and frequently it is carried out with a degree of freedom, having as a result label inconsistency. In other words, bad labeling and irrelevant information or limited information lead to inconsistent business process models, creating difficulties in the explanation, analysis and reusing of the model. The same situation also occurs when the same label is used for different elements or different labels is used for describing the same element [10].

Our purpose is to add semi-automated semantic annotations to business process elements, especially in activities (tasks and sub-processes) of business processes, as they are the main elements of a business process model, in order to augment business processes with concepts taken from a domain ontology clarifying the meaning of these process elements. The final product is a new semantically annotated business process model in which its activities have been tagged with linkable concept descriptions taken from a domain ontology, starting with the symbol "@" in business process diagram.

In order to match labels of business process activities with domain ontology concepts we measure the similarity between them, exploiting three similarity measures: (1) string similarities, (2) linguistic similarities and (3) syntactic similarities. In order to compute the string similarity degree we compare the number of common characters in the process element name and concept name, using different string algorithms. The linguistic similarity degree (natural language parsing) of the process element labels and of the concept names relies on a dictionary, called WordNet, to determine synonyms between them. In the same dictionary based also the syntactic similarity degree which detects homonyms and hyponyms, exploiting the context of names. Finally, a threshold filter determines whether a match is considered as confident or not. If the average similarity of a match is below the threshold then it is not considered as a match, otherwise it is and an annotation suggestion provided to the business designers for the semantic annotation of business process activities with concepts from an accurate domain ontology. Semantic annotations and the process which follows to find them are analyzed in details in chapter 4.

Finally, having created a basic ontology for BPMN in OWL format (more details in chapter 3) we express standardized queries in an ontology query language, called SPARQL language, using the Virtuoso repository. Moreover, the business designers have the opportunity to express their own queries if they have the knowledge.

The SeMFIS tool from the BOC Group organization has been extended in order to implement all the above features and processes. All the new functionalities of SeMFIS tool are presented in chapter 5.

1.3 Thesis Overview

The Thesis structure is as follows:

In chapter 2 we will recall the definitions of most popular Business Process Modeling Languages, Business Process Modeling Tools, ontologies, OWL/RDF, and semantic annotation. We will also refer the main notions of BPMN, as this language is used to our work. Furthermore, we will review the related scientific work done within the past years.

In chapter 3 we will present a basic ontology for BPMN 2.0 elements which will be used to querying the enrichment with semantic annotations business process models by means of Virtuoso Server.

Chapter 4 we will describe our approach for measuring String, Linguistic and Syntactic similarity between business process activity labels and domain ontology concept names. We aggregate these similarity measures to a combined similarity measure and filter it with a threshold to determine the semantic annotations of business process activities which will be suggested to business designers.

Application of our approach will be illustrated in chapter 5. In particular, this chapter focus on new functionalities of SeMFIS extended tool.

In chapter 6 the conclusions are drawn and an outlook on future research is presented.

Chapter 2

Background and Related Work

In this chapter, definitions about Business Process Modeling Languages, Ontologies, RDF/RDFS, OWL and Semantic Annotations are introduced. More emphasis are given to description of Business Process Modeling Notation (BPMN), as it is the language which is used in this work. Followed by a short presentation for some representative tools for creating business process models and finally, most recent related work are presented.

2.1. Business Process Modeling Languages

The business processes are conceptually modeled using various conceptual Business Process Modeling Languages (BPMLs). A comprehensive list with most popular Business Process Modeling Languages which have either future potential or are well-established in research range from Petri Nets (Petri 1962), Event-Driven Process Chain (EPC) (Scheer 2000) and UML Activity Diagram (AD) (Object Management Group 2004) to the Business Process Modeling Notation (BPMN) (Object Management Group 2004).

Despite their common aims, in [12] Van der Aalst distinguishes three language categories: (1) *Formal languages,* are based upon theoretical formalisms providing unambiguous semantics for describing business process models and allowing for analysis. In this category included Petri Nets. (2) *Conceptual languages,* do not characterized by the rigorous semantics of the formal languages. These languages are typically informal with some fuzziness in the modeling and do not allow analysis. However, these languages provide robust graphical notations and consequently enable convenient and intuitive modeling. For these reasons, they are preferred by business analysts in the initial phase of design business process model. In this category included EPC, UML AD, BPMN. (3) *Execution languages,* includes more technical languages that are concerned with business process execution, such as Business Process Execution Language (BPEL) and for this reason we do not conclude them in our research.

Petri Nets: in [7], Petri Nets are described as a widely accepted graphical language for the specification, simulation and verification of behavior of information systems.

A Petri Net is a directed bipartite graph consists of two types of nodes, places and transitions. Places represent conditions (possible states of the system), designed by circles. Transitions present events that may occur or actions which cause change of state, designed by rectangles.[13] The directed arcs describe which places are pre- and/or post-conditions for which transitions. No arc may connect a place to another place or a transition to another transition. It is one of several mathematical modeling languages for the description of dynamic systems.

Event Driven Process Chain (EPC): The EPC [13, 14, 19] is a modeling language for the graphical representation of a sequence of steps of a business process with the goal to be easily understood. It is a directed and connected graph, whose nodes are functions, events and logical connectors (AND/ OR/ XOR). Functions model the activities of a business process, while events are created by processing function or by actors outside of the model. Functions and events are related with connections (arrows) and logical connectors. Additional information like "document" or "role" complete the process description.

UML 2.0 Activity Diagram (AD): Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are designed for modeling business processes and flows in software systems. [13] Activity diagrams are constructed from a limited number of shapes, connected with arrows. Arrows run from the start towards the end and represent the order in which activities happen, showing the overall flow of control. The advantage [19] of such diagrams is that they are comprehensible by software engineers responsible for the implementation of business supporting components. Instead, the closeness to computing languages makes it difficult to be used by business analysts who do not have experience in software design.

Business Process Modeling Notation (BPMN): BPMN [15,16,17] is a state of the art graphical language for generating business process diagrams, which are based on a flowcharting technique tailored for creating graphical models of business process operations. The primary goal of BPMN [20, 19, 18, 17] is to provide a simple and intuitive notation that is readily understandable by all business users. In particular, the business analysts can create the initial drafts of the processes, technical developers responsible for implementing the technology that will perform those processes can easily and precisely convey the business analysts' ideas to technological implementation and finally business people can easily manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation. Another goal also is to enable portability of process definitions, so that users can take process definitions created in one vendor's environment and use them in another vendor's environment.

It is the mentioned combination of easiness of use with the precision of a formally well defined notation system which is responsible for the enormous success of BPMN, making it an international standard to model business processes. For this reason, we have selected the BPMN language for our research.

2.2. Main Notions of Business Process Modeling Notation (BPMN)

In our research, we use the BPMN 2.0 specification [20] which resolves BPMN 1.2 inconsistencies and ambiguities and extends the scope and capabilities of the BPMN 1.2 in: (1) formalizing the execution semantics for all BPMN elements, (2) defining an extensibility mechanism for both process model extensions and graphical extensions (3) refining event composition and correlation and (4) extending the definition of human interactions. It also provides multiple diagrams, which are designed for use by the people who design and manage BPMN and mapping them to an execution language of BPM Systems (Web Service Business Process Execution Language - WSBPEL 2.0).

A Business Process Diagram is made up of a set of graphical elements so as to be distinguishable from each other and to utilize shapes that are familiar to most designers. In [20, 16] is emphasized that one of the drivers for the development of BPMN is to create a simple and understandable mechanism for creating Business Process models, while at the same time being able to handle the complexity inherent to Business Processes. The approach taken to handle these two conflicting requirements was to organize the graphical aspects of the notation into specific categories. This provides a small set of notation categories so that the reader of a BPMN diagram can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look and feel of the diagram. The five basic categories of elements are:

1. **Flow Objects**: are the main graphical elements to define the behavior of a Business Process and consist of three core elements - events, activities and gateways.
   - 1.1. **Event**: is represented by a circle and is something that "happen" during the course of a business process. It effects the flow of the process and usually have a cause (trigger) or an impact (result). There are three types of Events, based on when they effect the flow, at the start of process (start event), during the process (intermediate event) or at the end of process (end event).

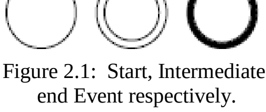


Figure 2.1: Start, Intermediate, end Event respectively.

   - 1.2. **Activity**: is represented by a rounded-corner rectangle and is a generic term for work that company performs in a process. An activity can be atomic or non-atomic (compound). The types of activities are Task and Sub-Process.

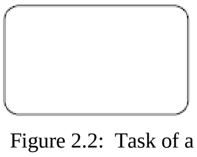


Figure 2.2: Task of a process

1.3. **Gateway**: is represented by the diamond shape and is used to control the divergence and convergence of sequence flow in a process. Thus, it will determine branching, forking, merging and joining of paths. Internal markers will indicate the type of behavior control.

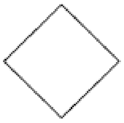


Figure 2.3: Gateway

2. **Data**: is represented with the four elements - data objects, data inputs, data outputs and data stores.

2.1. **Data Objects**: provide information about what activities require to be performed and /or what they produce. They can represent a singular or a collection of objects. Data Input and Data Output represent the same information for processes.

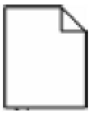


Figure 2.4: Data Object

3. **Connecting Objects**: used to connect flow objects to each other or other information in a diagram creating the basic skeletal structure of a business process. The connecting objects consist of four types - sequence flows, message flows, associations and data associations.

3.1. **Sequence Flow**: is represented by a solid line with a solid arrowhead and is used to show the order that activities will be performed in a Process.

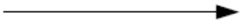


Figure 2.5: Sequence Flow

3.2. **Message Flow**: is represented by a dashed line with an open arrowhead and is used to show the flow of messages between two separate Process Participants (business entities or business roles) that send and receive them. In BPMN, two separate Pools in the Diagram will represent the two Participants.

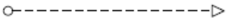


Figure 2.6: Message Flow

3.3. **Association**: is represented by a dotted line and is used to link information and Artifacts with flow objects.

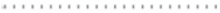


Figure 2.7: Association

3.4. **Data Association**: is represented by a dotted line with a line arrowhead and is used to associate data objects with flow objects.

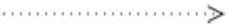


Figure 2.8: Data Association

4. **Swimlanes**: is a mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities. BPMN supports two types of swimlane objects - pool and lane.

4.1. **Pool**: represents a Participant in a Process. It is also acts as a graphical container for partitioning a set of activities from other Pools.

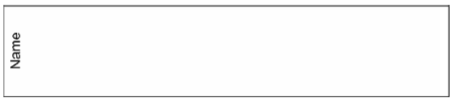


Figure 2.9: Pool

4.2. **Lane**: is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organize and categorize activities.

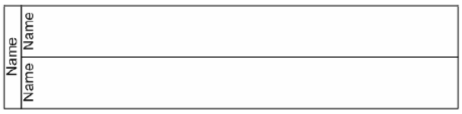


Figure 2.10: Lane

5. **Artifacts**: are used to provide additional information about the process making it more readable. There are two standardized types of Artifacts - group and text

annotation - but modelers of modeling tools can add as many artifacts as necessary allowing some flexibility to extend the basic notation.

5.1. **Group**: is represented by a rounded corner rectangle drawn with a dashed line. It is a grouping of graphical elements that can be used for documentation or analysis purposes, but does not affect the Sequence Flow.

Figure 2.11: Group

5.2. **Text Annotations**: are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram.

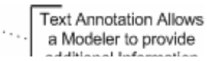


Figure 2.12: Text Annotation

2.3. Ontologies

The term ontology [22] has its origin in philosophy and specifically, the word element "*onto-"* comes from the Greek "*ὤν*", "*ὄντος*" which mean "*being*", "*that which is*". In computer science, ontologies are developed to provide a machine-readable semantics of information sources that can be communicated between different agents (software and humans). Instead of "ontology" we now speak of "an ontology".

Many definitions of ontologies have been given in the last decade, but one that best characterizes the essence of an ontology is based on the T.R. Gruber definition (1993), later refined by R. Studer (1998): *An ontology is a formal and explicit specification of a shared conceptualisation*. According to [23] the meaning "*conceptualisation*" refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon. "*Explicit*" means that the type of concepts used and the constraints on their use must explicitly defined. "*Formal*" refers to the fact that the ontology should be machine readable. Hereby different degrees of formality are possible. Finally the meaning "*Shared*" reflects the notion that an ontology captures consensual knowledge, that is, it is not restricted to some individual, but accepted by a group.

Another core meaning of an ontology, given in [22], is that it is a model for describing formally a domain of discourse that consists of a set of terms (that is the vocabulary of ontology), relationships between these terms and an inference mechanism for it. The notion "*terms*" denote important concepts of the domain (that is classes of objects) while the notion "*relationships*" includes hierarchies of classes. In particular, a hierarchy specifies a class C to be a subclass of another class C' if every object in C is also included in C'. For example, speaking about a university camp, staff members, students, courses are some important concepts while the fact that all academic staff are staff members or that a postgraduate are student characterizes an hierarchy for the university people. Apart from subclass relationships, ontologies may include information about: (1) *Properties* such as "a student attends courses", (2) *Value Restrictions* such as "only faculty members can teach courses", (3) *Disjointness Statements* like the statement "faculty and general staff are disjoint" and (4) *Specification of Logical Relationships between Objects* like "every department must include at least ten faculty members". Thus, ontologies are a structured source of knowledge (a taxonomy) permitting the standardization of concepts, supporting the interoperability at the semantic level and reasoning [10].

The ontologies are classified in different types depending on their generality of level. Among others, the following ontology types can be distinguished [23]: (1) *Upper level ontologies* or *General ontologies* capture general knowledge about the world providing basic notions and concepts for things like events and states. (2) *Domain ontologies* capture the concepts of a particular area of interest or a specific topic, for example digital domain or medical domain.

In [24] is referred that Ontologies can also be expressed in Description Logics (DL), a well-known family of knowledge representation formalisms. In particular, an ontology can be regarded as a typical DL knowledge base which consists of two components: a "TBox and a "ABox". TBox represents the background knowledge and the knowledge about the terminology relevant for the described domain, including the concepts, their properties and their relations as a set of asserted axioms while ABox represents the individuals that are instances of concepts of the ontology in the form of membership statements.

The Web, World Wide Web Consortium (W3C) has proposed a DL-based web ontology language: the OWL. It is a formal description for creating, publishing and distributing ontologies [19]. It provides a set of vocabulary as constructs, enabling people to define concepts, properties, individuals, and their relations. Typically, a property in OWL can be distinguished in two categories: (1) data type property which allows people to describe specific attributes of a concept, such as "date of birth", "age of person" and (2) object property which enables people to link two concepts with a semantic relation, like “teaches” between “professor” and “student”. Corresponding to the notions of TBox and ABox in DL, ontology encoded in OWL can also be partitioned into two parts: ontology schema and ontology data. Definitions of concepts, properties and their relations in the owl file(s) are treated as ontology schema. Instances of these concepts (that is individuals) are treated as ontology data [24].

2.4. RDF/ RDFS

RDF (Resource Description Framework) [22, 65] is essentially a graph-based data model for the web. It is used to represent information about resources on the web, as well as for things that can be identified on the web, even when they cannot be directly retrieved on the web, like a person. The main intention of RDF data model is to be used for situations in which information about web resources needs to be machine-accessible and machine-processable, meaning that it needs to be processed by applications, rather than being only displayed to people.

Its basic building block [22,62, 65] is a triple, called statement, which is a triad (**s, p, o**) where **s** is called subject, **p** is called property and **o** is called object. The *subject* represents a resource, meaning a "thing" that we want to talk about, e.g. books, people, places, animals and so on. It is a URI (a Universal Resource Identifier) or a Blank Node, both of which denote resources. Resources indicated by Blank Nodes are called anonymous resources and they are not directly identifiable by the RDF statement. A URI can be a URL (Unified Resource Locator, web address) or some other kind of unique identifier. The basic URI syntax consists of a URI scheme name, e.g. http, mailto, file, followed by a colon character and then by a scheme specific part, as shown below:

<URI scheme name> : <scheme specific part>

In general, an identifier does not necessarily enable access to a resource and are not limited to identify things that have network locations. It can also identify diverse objects, such as telephone numbers and ISBN numbers. *Properties* are a special kind of resources. They describe relations between resources, more precisely between the subject and the object of the statement, for example "has_age" or "has_title". An RDF statement offers only binary predicates (properties). We can think an RDF triple (x, P, y) as a logical formula P(x,y) where the binary predicate P relates the object x to the object y. Properties are also identified by URIs. An *object* could also be a resource identified by a URI, a Blank Node or a literal. There are two kinds of literals: (1) atomic values (strings) which have a lexical form and optionally a language tag, e.g. "25", "name"@en and (2) RDF typed literals which are formed by pairing a string with a URI that identifies a specific datatype, e.g. "25"^^http://www.w3.org/2001/XMLSchema#integer. They just indicate explicitly what data type would be used to interpret a given literal. The "^^" notation indicate the type of the given literal. The data types, which are used most widely in RDF documents, are predefined by XML schema, including integers, Booleans, floats, times and dates.

A set of RDF triples forms an RDF graph. It is a directed graph with labeled nodes and arcs. The arcs are directed from the "subject" resource of the statement to the "object" value of the statement and represent relations between the nodes. This kind of graph is actually a semantic network. RDF statements are expressed most widely using the following machine-readable formats: RDF/XML, Turtle, N3, Json, RDFa (embedded in HTML pages) [22, 65].

RDF Schema (**RDFS**) [63] provides a data-modeling vocabulary for RDF data. As referred at [64], it is a set of classes with certain properties using to extend RDF data model, providing basic elements for the description of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources. A class [22] can be defined as a set of resources. An individual object that belongs to a class is called instance of that class. Classes are themselves resources. The relationship between instances and classes in an RDF statement is expressed using the property "*rdf: type*".

RDFS also establishes relationships between the classes themselves defining a hierarchy of classes. The property "*rdfs: subClassOf*" is used to state that one class is subclass of another class. If a class A is a *subclass* of a class B, then all instances of A will also be instances of B. The term super-class is used as the inverse of subclass. An important point of RDFS is that it fixes the semantics of "is a subclass of". This means that it is not up to an application to interpret "is a subclass of", its intended meaning must be used by all RDF processing software. Hierarchical relationships can also be established between properties. This is done with the property "*rdfs: subPropertyOf*" which denotes that P is a subproperty of Q if Q(x,y) whenever P(x,y). Except of the properties "*rdf:type*", "*rdfs:subClassOf*" and "*rdfs:subPropertyOf*", which have already been described, other main RDFS constructs are the following classes and properties.

The classes are:

- *rdfs:Class*, the class of all classes.
- *rdfs:Resource*, the class of all resources.
- *rdfs:Property*, the class of all properties.
- *rdfs:Literal*, the class of literal values such as strings and integers.

The additional properties are:

- *rdfs:domain* for a property P, specifies the class of those resources that may appear as subject in an RDF triple whose predicate is that property P.
- *rdfs:range* for a property P, specifies the data type or the class of those resources that may appear as object in an RDF triple whose predicate is that property P.

2.5. Web Ontology Language: OWL

The expressivity of RDF/RDFS [22, 66] is too weak to describe resources in sufficient detail. RDF is limited to binary predicates and RDF Schema is limited to subclass and subproperty hierarchy, domain and range restrictions and instances of classes. However, a number of other features are missing for describing the semantic of knowledge precisely. Some of the most important features are:

<u>Localized range and domain constraints</u>: In RDFS we cannot declare range restrictions that apply to some classes only. For example, we cannot say that the range of property "eat" in cows is "plants", while for other animals may also be "meat".

<u>Disjointness of classes</u>: In RDFS we can only state subclass relationships. For example, we can say that "female" is subclass of "person". But we cannot say that "female" is disjoint with "male".

Existence/ Cardinality constraints: In RDFS we cannot declare restrictions on how many distinct values a property may take. For example, we cannot say that a person has exactly two parents or that all instances of "person" have a mother that is also a person.

Boolean combinations of classes: In RDFS we cannot build new classes by combining other classes using union, intersection and complement. For example, we may want to define the class "person" to be the disjoint union of the classes "male" and "female".

Special characteristics of properties: In RDFS we cannot define that a property is *transitive*, e.g. the property "isPartOf", *inverse*, e.g. the property "hasPart" is inverse of "isPartOf", *unique*, e.g. the property "isMotherOf", or *symmetrical*, e.g. the property "touches".

Reasoning support: RDFS is difficult to provide reasoning as there are no "native" reasoners for non-standard semantics.

Thus, there is the need for an ontology language that is more powerful and richer than RDF Schema. A language which allows users to write explicit and formal conceptualizations of domain models, offering the above features and more. A such language must keep the following requirements:

well-defined syntax: It is a necessary condition for machine-processing of information.

formal semantics: There should be no doubt about the meaning of knowledge. This means that the semantics does not refer to subjective intuitions, nor is it open to different interpretations by different people or machines.

As far as ontological knowledge is concerned, the formal semantics allow people to reason about:

- Class membership: If x is an instance of class A and A is a subclass of B, then we can infer that x is an instance of B.
- Equivalence of classes: If a class A is equivalent of class B and class B is equivalent of class C, then A is equivalent to C, too.
- Consistency: Suppose we have declared x to be an instance of class A and that A is a subclass of B∩C, A also is a subclass of D, and B and D are disjoint. Then we have an inconsistency because A should be empty but has the instance x.
- Classification: If we have declared that certain property-value pairs are a sufficient condition for membership in a class A, then if an individual x satisfies such conditions, we can conclude that x must be an instance of A.

Semantics also is a prerequisite for efficient automated reasoning support. The latter is important because it allows to someone without missing time to:

- check the consistency of the ontology and the knowledge in general.
- check for unintended relationships between classes.
- automatically classify instances in classes.

After a number of researches, W3C Organization defined the OWL (Web Ontology Language) as the standard ontology language of the Semantic Web. A language that can be supported by efficient reasoners while being sufficiently expressive to express large classes of ontologies. OWL can be consider as an extension of RDF Schema, in the sense that OWL builds upon RDF and RDF Schema. Instances are defined using RDF descriptions and most RDFS modeling primitives (rdfs:subClassOf, rdfs:Class, rdfs:domain, rdfs:range etc.) are used.

OWL [67] provides three increasingly expressive sublanguages: (1) OWL Lite is intended for users who need a classification hierarchy and simple constraints. For example, it only permits cardinality values of 0 or 1. (2) OWL DL is intended for users who want the maximum expressiveness retaining at the same time computational completeness and decidability, meaning that all computations will finish in finite time. It includes all OWL and RDF language constructs, restricting how they may be used. For example, while a class may be a subclass of many classes, a class cannot be an instance of another class. It is optimized kind of OWL language for reasoning and knowledge modeling. (3) OWL Full is intended for users who want the maximum expressiveness and the syntactic freedom of RDF. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full also allows an ontology to augment or change the meaning of the pre-defined RDF or OWL vocabulary. The main disadvantage of OWL Full is that it does not conclude complete and efficient reasoning support.

The main constructs of an OWL Ontology [22, 67] are:

Header: It concludes the "rdf:RDF" element which specifies a number of namespaces. For example:

```
<rdf:RDF

        xmlns:owl ="http://www.w3.org/2002/07/owl#"

        xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

        xmlns:xsd ="http://www.w3.org/2001/XLMSchema#">
```

and the "owl:Ontology" element, which contains comments, version and inclusion of other ontologies. For example:

```
<owl:Ontology rdf:about="">
```

```
    <rdfs:comment>An example OWL ontology</rdfs:comment>
    <owl:priorVersion
        rdf:resource="http://www.mydomain.org/uni-ns-old"/>
    <owl:imports
        rdf:resource="http://www.mydomain.org/persons"/>
    <rdfs:label>University Ontology</rdfs:label>
</owl:Ontology>
```

Classes and class hierarchy: Classes are defined using the "owl:Class" element. There are also two pre-defined classes, "owl:Thing" and "owl:Nothing", where the first one is the most general class, containing everything and the last one is the more specific, the empty object class. We can also define relations between classes and specifically: (1) Disjointness of classes with the element "owl:disjointWith" and (2) Equivalence of classes using the element "owl:equivalentClass".

Properties: There are two kinds of properties: (1) Object properties, which relate objects to other objects and (2) Data type properties, which relate objects to datatype values. It adopts XML Schema data types e.g. integers, floats, strings, Boolean, time, date, etc.

Property restrictions: They deal with how properties can be used by instances of a class. All restrictions are used within the context of an "owl:Restriction" element. The "owl:onProperty" element, which is contained on "owl:Restriction" element, indicates the restricted property. OWL defines two types of restriction declarations. One type limits which values can be used by property and are: (1) "owl:allValuesFrom", which is used to specify the class of possible values the property specified by "owl:onProperty" can take, (2) "owl:hasValue", which states a specific value that the property specified by "owl:onProperty" must have and (3) "owl:someValuesFrom", which does not restrict all values of the property specified by "owl:onProperty" to be instances of the same class. The other type of restriction declarations defines cardinality restrictions and are: (1) "owl:minCardinality", which states at least one individual by the property specified by "owl:onProperty", (2) "owl:maxCardinality", which states at most one individual by the property specified by "owl:onProperty" and (3) "owl:Cardinality", which specifies a precise number.

Characteristics of properties: OWL defines some special properties applied to object properties which are: (1) "owl:TransitiveProperty", which defines a transitive property, e.g. "is greater than", (2) "owl:SymmetricProperty", which defines a symmetric property, such as "has same grade as", (3) "owl:FunctionalProperty", which defines a property that has at most one unique value for each object, such as "age", "height" and (4) "owl:InverseFunctionalProperty", which defines a property for which two different objects cannot have the same value, such as "hasAFM".

Boolean combinations: OWL supports Boolean combinations - union, intersection and complement - of classes using the "owl"unionOf", "owl:intersectionOf" and "owl:complementOf" elements respectively.

Enumerations: The "owl:oneOf" element is used to define a class by listing all its elements.

Individuals: Instances of classes are declared as in RDF, using the "rdf:type" element. OWL does not adopt the unique-names assumption, which means that if two instances have a different name or ID, it does not imply that they are different individuals. As a consequence, in order to declare that two individuals are equal, OWL uses the element "owl:sameAs", while to declare that two individuals are unequal, it uses the element "owl:differentFrom". If we want to declare that a number of individuals are mutually distinct, the OWL uses the "owl:AllDifferent" element. It is used in conjunction with the "owl:distinctMembers" element to state that all members of a list are distinct and pairwise disjoint.

2.6. Semantic Annotation

Generally speaking, the term "Annotation" implies to attach data to other piece of data adding more specific information (that is a description) about it [25]. The term "Semantic Annotation" is a clear and easy to understand by both, human and machines, specification which is used to add meaning taken from an ontology to a specific data. We can distinguish three categories of semantic annotations: (1) Manual annotations performed by one or more users allowing them to choose manually the annotation which will be added to a specific object, (2) Semi-Automatic annotations based in automatic suggestions by tool allowing to user to choose one annotation by the suggested ones or manually annotated the specific object and (3) fully automatic annotations based in exclusive proposal by tool without manual intervention.

In [25] a formal definition is given according to which an Annotation is a quadruple (a_s, a_p, a_o, a_c) where a_s is the subject of the annotation (the annotated data) a_o is the object of the annotation (the annotating data) a_p is the predicate (the annotation relation) that defines the type of relationship between a_s and a_o, and a_c is the context in which the annotation is made.

2.7. Business Process Modeling Tools

In order to be able business analysts to design the graphical representation of business process models, a business process modeling tool is needed. Some of these tools which use BPMN 2.0 Specification are Activiti, Eclipse BPMN2 Modeler, Camunda, Adonis CE, Maestro for BPMN, SeMFIS.

Activiti [27] is a light-weight workflow and Business Process Management (BPM) Platform targeted at business people, developers and system administrators. It sponsored by enterprise content management giant Alfresco. Its core is a super-fast and rock-solid BPMN 2.0 process engine for Java. It is open-source and distributed under the Apache license. Specifically it is licensed under the Apache License 2.0 to encourage widespread usage and adoption of the Activiti BPM engine and BPMN 2.0. Activiti runs in any Java application, on a server, on a cluster or in the cloud because it is just a jar file. The main components of activiti which are combined to form a complete solution from BPMN are showed in the following Figure:

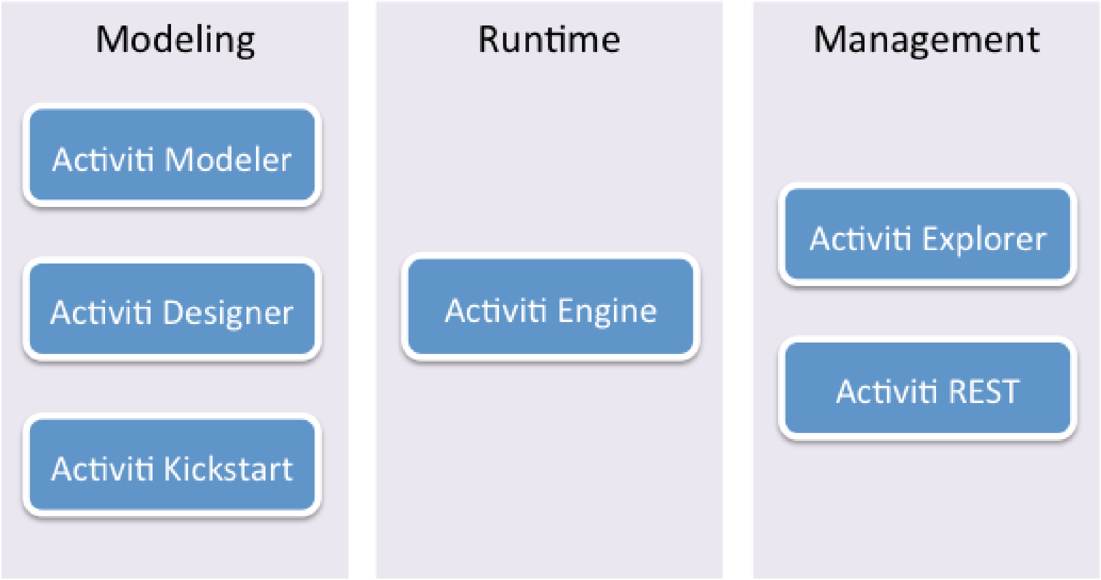
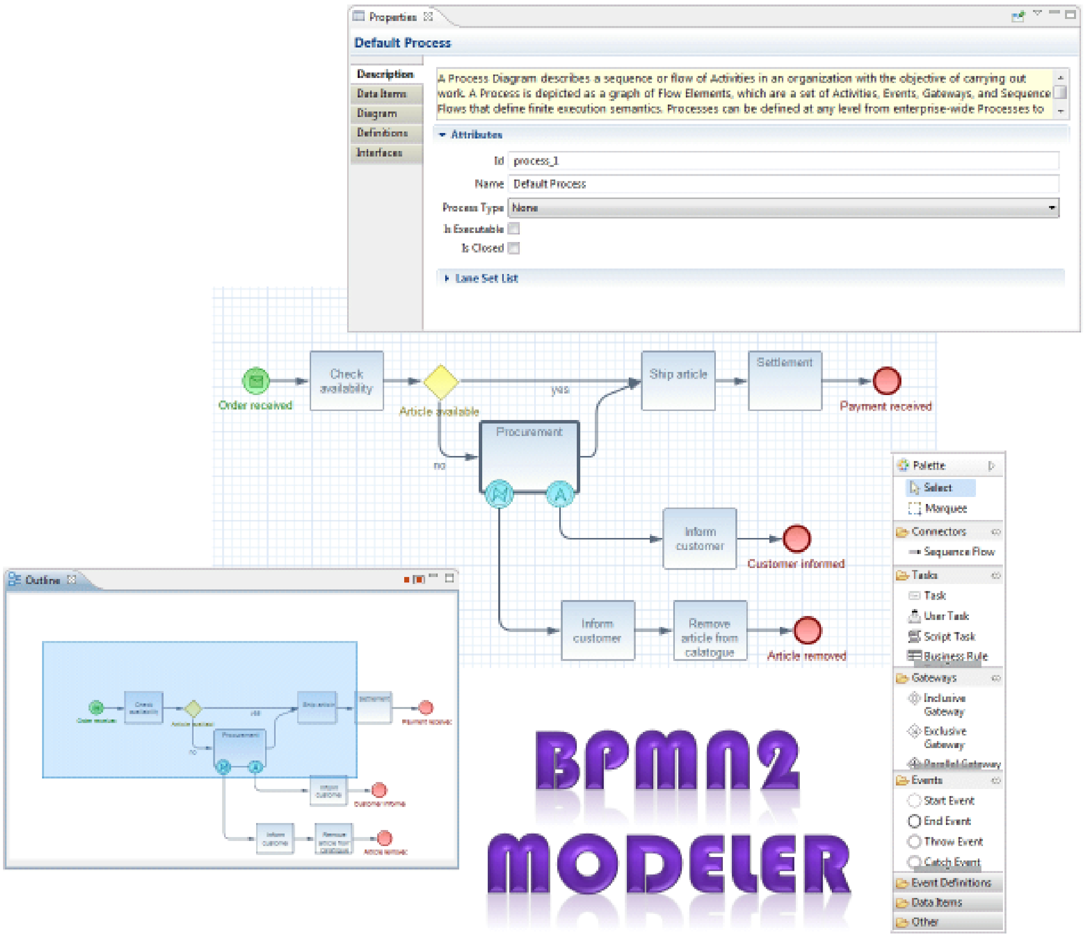


Figure 2.13: Activiti Components

Activiti Modeler can be used to author BPMN 2.0 compliant processes graphically using a browser. The process files are stored by the server in a database model repository while the **Activiti Designer** is an Eclipse plugin which allows you to model BPMN 2.0 processes within your IDE-environment. Although it supports Modeler, Simulation and Execution, it has two drawbacks: (1) data elements are not supported and (2) it permits limited supported formats, that is read and save internally in BPMN format without exporting capabilities.

Eclipse BPMN2 Modeler [28] is a graphical modeling tool for authoring business processes. The primary goal of BPMN2 Modeler was to provide a graphical workflow editing framework, which can be easily customized for any BPMN 2.0 compliant execution engine. It was built by eclipse org. as a mission of the eclipse Service Oriented Architecture (SOA) project and it distributed under the Eclipse Public License 1.0. It is a cross-platform product meaning that it runs both on windows and on Linux, as well as on other platforms. The BPMN2 Modeler is built on the Eclipse Plug-in Architecture and provides several extension points for customizing the editor's appearance and behavior. Nevertheless, at this time, it provides incomplete support for data elements which means that it does not support data store, data input/output elements and data collection. Supported formats for reading is BPMN2 and for writing are BPMN2, BMP, GIF, JPG, PNG, PDF.



Default Process

A Process Diagram describes a sequence or flow of Activities in an organization with the objective of carrying out work. A Process is depicted as a graph of Flow Elements, which are a set of Activities, Events, Gateways, and Sequence Flows that define finite execution semantics. Processes can be defined at any level from enterprise-wide Processes to

Attributes

Id: process_1

Name: Default Process

Process Type: None

Is Executable □

Is Closed □

Lane Set List

Figure 2.14: Screenshot of BPMN2 Modeler tool

The core of **Camunda BPM** [29] is an execution engine for BPMN, CMMN and DMN. It is lightweight and requires less than 3MB of disk space. It can run in any Java Virtual Machine (JVM). It is distributed under the Apache License 2.0 and runs on Windows, Linux and Mac.

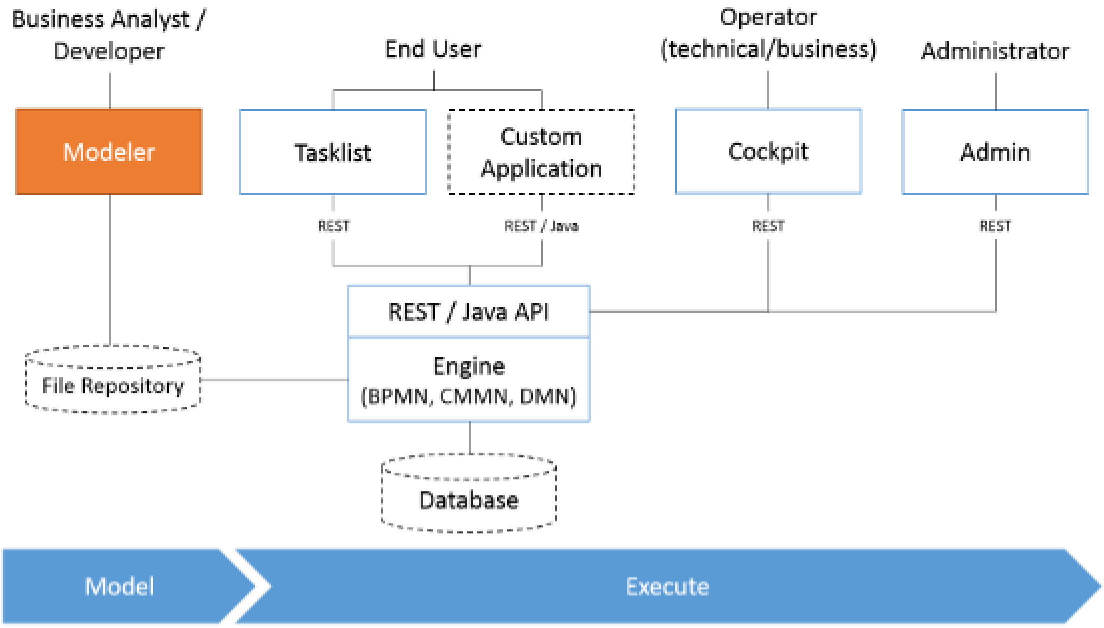
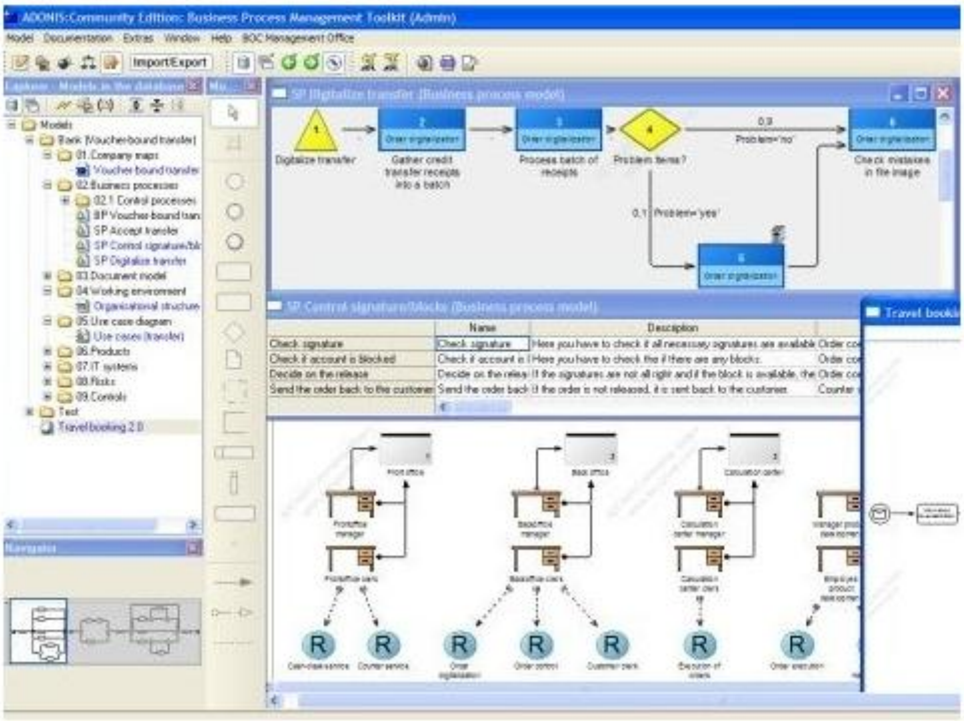


Figure 2.15: Camunda BPM Architecture

Camunda Modeler is a desktop application for editing BPMN process diagrams supporting BPMN 2.0 Specification. It is very easy to use, which means that business analysts can use it as well as developers, working on the same diagrams. Besides the visual modeling, Camunda Modeler also allows you to edit all properties that are necessary for the technical execution. Since Camunda Modeler works directly on the BPMN and DMN XML files, developers can easily combine it with their preferred IDE (for example Eclipse, Netbeans). BPMN 2.0 modeling based on bpmn.io which is probably the most awesome modeling framework in the known universe. However, it partially supports data elements which means that it does not support data store, data input/output and data collection. It also does not support BPMN2 comments.

Adonis Community Edition (CE) [31] is the free version of Adonis BPM tools, without any time limitation. Most of the BPM software are quite expensive, but Adonis community edition is completely free for both personal as well as commercial uses. It is developed by the BOC Group which was founded as a spin-off of the University of Vienna. It lets you model business processes easily, supporting BPMN 2.0 Specification. It also allows analysis, simulation, evaluation and Sharing BPM models. Unfortunately, it runs only on windows platform.



	Name	Description	
Check signature	Check signature	Here you have to check if all necessary signatures are available	Order co
Check if account is blocked	Check if account is I	Here you have to check the if there are any blocks.	Order co
Decide on the release	Decide on the relea	If the signatures are not all right and if the block is available, the	Order co
Send the order back to the customer	Send the order back	If the order is not released, it is sent back to the customer.	Counter

Figure 2.16: Adonis CE Business Process Management Toolkit

As far as **Modeling** is concerned, Adonis CE lets you model your entire organization: processes, products, resources and see how they interrelate. It is quite easy in terms of use, which makes it quite easy for even new BPM modelers to quickly start modeling. Models can be saved as HTML, or even embedded in Word documents and presentations and general it provides options to export them in different formats such as ADL, XML, XPDL. It is also quite good in analyzing those models to find process bottlenecks and other inefficiencies in the system, as well as it provides simulation option so as to be able to find costs of processes, staff requirements, bottlenecks and other inefficiencies. Adonis CE stores all the model data in SQL Server free edition that comes with Adonis download and it is easy in installation.

Maestro for BPMN [61] is a SAP research modeling tool which has already been extended by Born et al. to enable semantic annotation of the business processes (a fragment of process model using Maestro for BPMN tool is shown in Figure 2.17). For this reason, it makes use of the sBPMN ontology from SUPER project. Specifically, if a new BPMN task is created on the drawing pane, an instance of the concept "Task" is created in the in-memory working ontology, enabling reasoning over it. More details on how they have achieved semantic annotations to business process models are presented in the "related work" section.

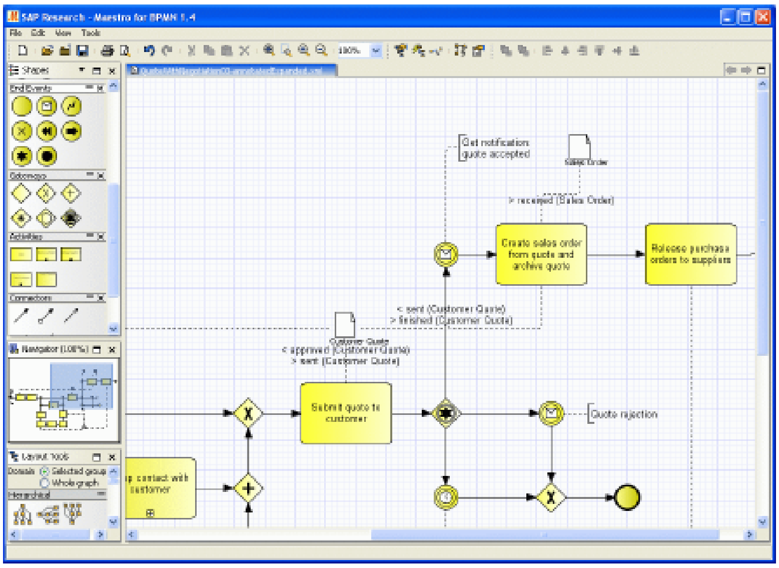
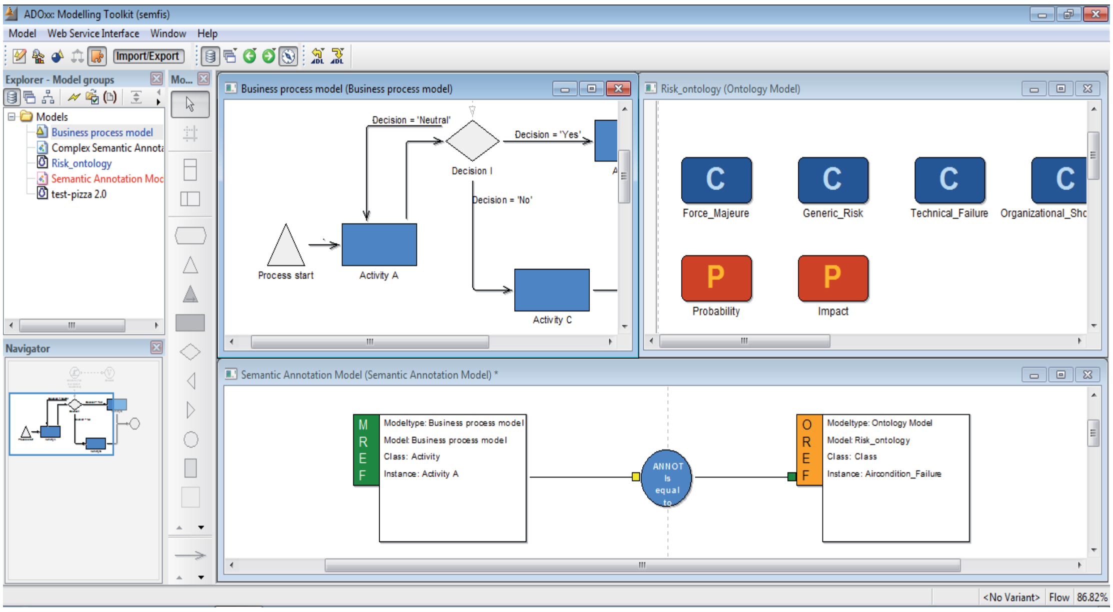


Figure 2.17: A fragment of a process model in Meastro for BPMN tool

Although it supports the graphical representation of business process models using BPMN specification, it also supports ontologies expressed in WSML/WSMO format, whereas we make use of OWL format. In addition, it is not freely available for using.

SeMFIS [30] means Semantic based Modeling Framework for Information Systems and it is a flexible engineering platform for semantic annotations of conceptual models that supports the representation and analysis of annotations with ontologies. SeMFIS has implemented using the Microsoft Windows-based ADOxx meta modeling platform which is professionally developed by BOC Group, a spin-off of the University of Vienna, and it has been on the market for more than fifteen years. Thus, SeMFIS can be easily added to the large variety of other modeling methods based on this platform or used as an additional service for other tools. The standard installation of SeMFIS is connected with the relational database of Microsoft SQL Server. It is freely available for use via the OMiLAB.org website at http://www.omilab.org/web/semfis.



Modeltype: Business process model
Model: Business process model
Class: Activity
Instance: Activity A

Modeltype: Ontology Model
Model: Risk_ontology
Class: Class
Instance: Aircondition_Failure

Figure 2.18: SeMFIS Model Editors for a Business Process Model, an OWL Ontology Model and a Semantic Annotation Model (from left to right, top to bottom)

SeMFIS intention was to provide a link between the field of conceptual modeling (like a business process model) and the field of ontologies. SeMFIS does not require a specific type of modeling language or the modification of an existing modeling language because of the decoupling of the semantic annotations in separate semantic annotation models, as shown in Figure 2.18 on bottom model for the creation of which the top models, business process model on the left and owl ontology on the right, contributed. The most important features are provided by SeMFIS platform are: *Model Editors*, *Scripting and Analysis functionalities*, *import and export interfaces*.

For the graphical representation of any conceptual model which is supported by SeMFIS, business process models, semantic annotations and any supported ontology type, SeMFIS has available its own **model editor**, as shown in Figure 2.18. Via the **scripting functionality**, statements in the domain-specific ADOscript language can be executed in order to extend SeMFIS functionalities. As far as **analysis component** is concerned, SeMFIS uses it in order to express queries in AQL format (ADOxx Query Language). Finally, SeMFIS provides **import and export interfaces** for exchanging model information in different file formats. However, only XML Import/Export interface is used to exchange information from arbitrary model types and ADL Import/Export Interface to exchange information with other ADOxx based tools that do not offer an XML interface.

In our work, we select the SeMFIS tool for the following reasons: (1) It has been implemented using the freely available ADOxx meta modeling platform and it is supported by the freely available AdoScript language for extending its functionalities. (2) It is a flexible platform which can be used to create both business process models (supporting BPMN 2.0 Specification) and Ontologies (supporting OWL language) using any time its own model editor. (3) As far as business process models are concerned, it already has a "semantic annotation" attribute on the objects' notebook which can be added manually on objects by the user. (4) Nevertheless it has a separate semantic annotation model type for linking objects of business process models with concepts of ontologies, it is useful only for expert users. Most of the times, business designers do not know about ontology technologies, so it is a useless functionality for them.

For all these reasons, exploited the "semantic annotation" attribute and the freely available use of ADOxx meta modeling platform, as well as the freely available AdoScript documentation, we use the SeMFIS tool in order to present our semi-automatic suggestions for semantic annotations on activity labels of business process models creating at the same time a new semantic annotated business process model so as not to modify existing ones. More details about extended functionalities are presented in chapter 5.

2.8. Related Work

The idea of adding semantic annotations to business process models is not new and has already been proposed by several authors, for different process languages and goals. We can roughly classify the existing proposals into two big groups. The first group consists of those who add semantic annotations to specify the dynamic behavior exhibited by a business process [5, 60]. The second group is composed of those who add semantic annotations to specify the meaning of process elements in order to improve the automation of business process management [2, 3, 4, 7, 8, 14, 36, 37]. In our work we follow the second perspective in order to give a more precise and comprehensive meaning in the activity labels of business process models and to allow the use of semantic technologies, like querying, on business processes via semantic annotations.

In [5], Markovic and Pereira present an expressive formalism for describing business process models to support reuse of existing business fragments during modeling. They distinguish two main aspects of a process description: dynamic and static aspect. As for the dynamic aspect, they have selected to use process algebra, the π-calculus, for capturing the behavior of the process, that is process control flow. Within the static aspect of the process description they want to describe other workflow perspectives, e.g. organizational and informational perspectives. Specifically, they want to describe processes in terms of their input/output data, business function, business domain, organizational roles which perform certain process parts. For this reason, they use WSML (Web Service Modeling Language) as a representation language for the ontologies that capture static aspects of the process model description. Finally, they propose the Business Process Ontology (BPO), which captures both dynamic and static aspects of a process model description. The concepts in BPO are visualized in Figure 2.19, using WSMO (Web Services Modeling Ontology) Studio.

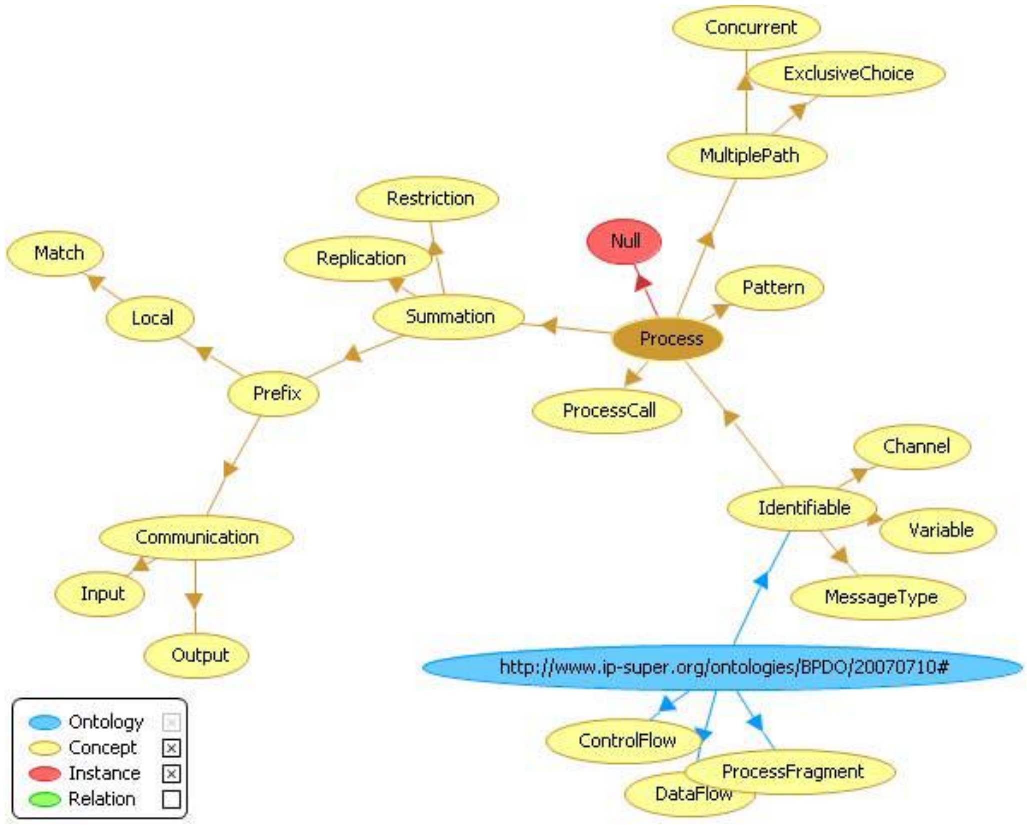


Figure 2.19: Concepts of Business Process Ontology (BPO)

In order to integrate behavioral with other workflow perspectives, BPO imports concepts from several other ontologies, forming the ontology framework shown in Figure 2.20.

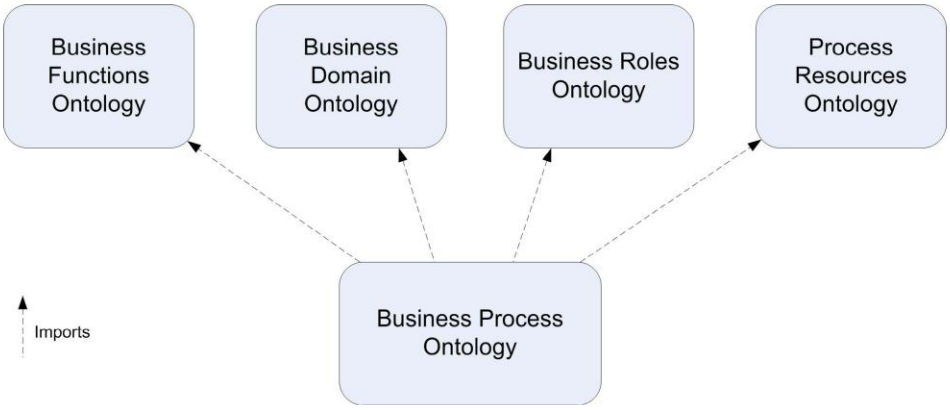

Business Functions Ontology
Business Domain Ontology
Business Roles Ontology
Process Resources Ontology
Imports
Business Process Ontology

Figure 2.20: Ontology Framework

Imported ontologies describe concepts used to create semantic annotations for concrete process definitions. Specifically, for describing the functional perspective, they have designed the Business Functions Ontology. For describing the domain (e.g. product area, client area, etc) inside the organization where the process is used, they designed the Business Domain Ontology. Business Roles Ontology includes concepts representing roles in the organization e.g. manager, engineer, secretary, etc. while Process Resources Ontology describes the resources (e.g. documents, systems, machines) which are required to operate the activities in processes. Finally, the semantic annotation of processes is done by five relations as shown in Figure 2.21: hasBusinessGoal, hasBusinessFunction, hasBusinessDomain, hasBusinessRole and hasProcessResource. These semantic annotations can be used for various querying and reasoning purposes e.g. finding process fragments, verification, execution, etc.

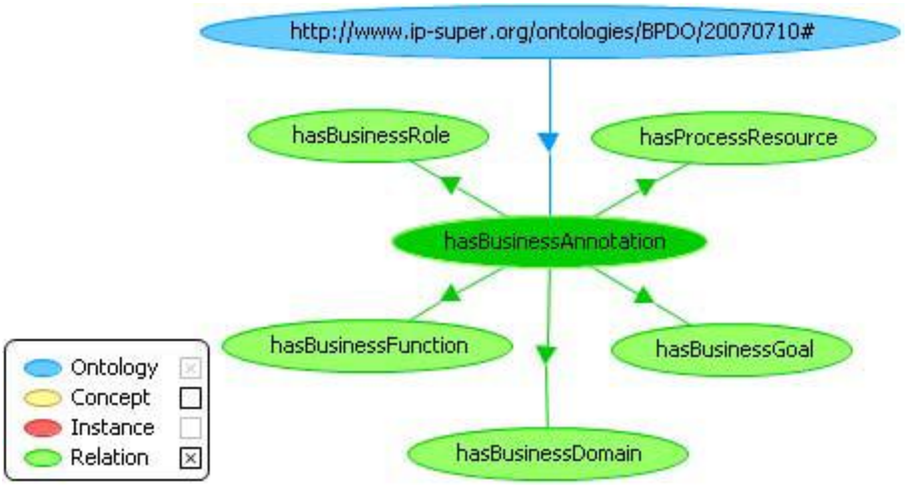


Figure 2.21: Business Annotations

In [60], Smith and Proietti propose a rule-based framework for reasoning about process-related knowledge expressed by using standards for business process modeling like BPMN specification and ontology definition like OWL language. In order to present the behavioral semantics, they follow an approach inspired to the Fluent Calculus, a well-known calculus for action and change. In the Fluent Calculus, the state of the world is represented as a collection of fluents, which means terms representing atomic properties that hold at a given instant of time. A fluent is an expression of the form f(a_1, ..., a_n) where f is a fluent symbol and a_1, ..., a_n are constants of variables. They define semantic annotations for modeling the behavior of individual process elements in terms of preconditions under which a flow element can be executed and effects on the state of the world after its execution. Preconditions and effects called functional annotations and can be used to model input/output relations of activities with data items, which are the standard way of representing information storage in BPMN diagrams. Specifically, a precondition specifies the status a data item must possess when an activity is enabled to start and is formulated by means of the relation: pre(A,C,P), which specifies the fluent expression C, called enabling condition, which must hold to execute an element A in the process P. An effect specifies the status of a data item after having completed an activity and is formulated by means of the relation: eff(A, E^-, E^+, P), which specifies the set E^- of fluents, called negative effects, which do not hold after the execution of A and the set of fluents E^+, called positive effects, which hold after the execution of A in the process P. They assume that E^- and E^+ are disjoint sets.

Both of the above works [5, 60] are focused on dynamic behavior of business process elements where we have not researched at all in this work.

In [14], Thomas and Fellmann consider the problem of enhancing individual model elements of event-driven process chains (EPC) with semantic annotations using concepts of a formal ontology. They propose a framework which joins process model and ontology by means of properties (such as the "semType" of a process element). Specifically, they propose a multi-level approach which comprises an ontology level, a metadata level and a model level, as shown in Figure 2.22. Metadata is generated from models. This metadata contains references to the model elements of the initial model, as well as to the concepts of the ontology. Ontologies and metadata are interdependent. Concepts from the ontology are used in the metadata to specify the meaning of the labels of the model elements. The ontologies used must contain the required concepts or they must be added to the ontologies in the course of the creation of the metadata.

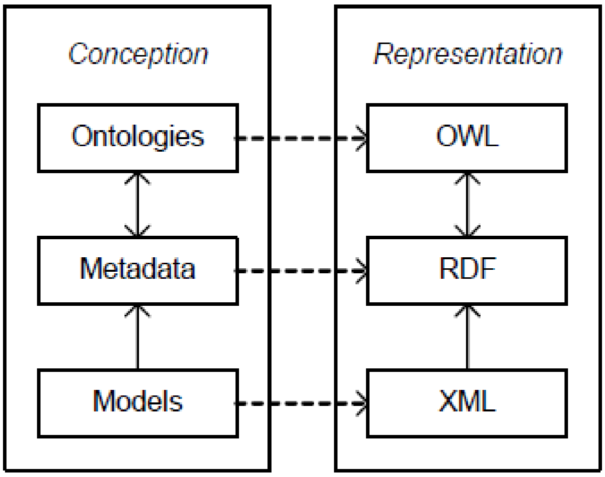


Figure 2.22: Framework for the semantic annotation of business process models

In the Figure 2.23 is presented the linkage of an ontology and the EPC model elements instances of "customer order processing" model which is accomplished by the usage of "semType" properties. These properties specify the semantics of an EPC model element through a relation to an ontology instance with formal semantics defined by the ontology.

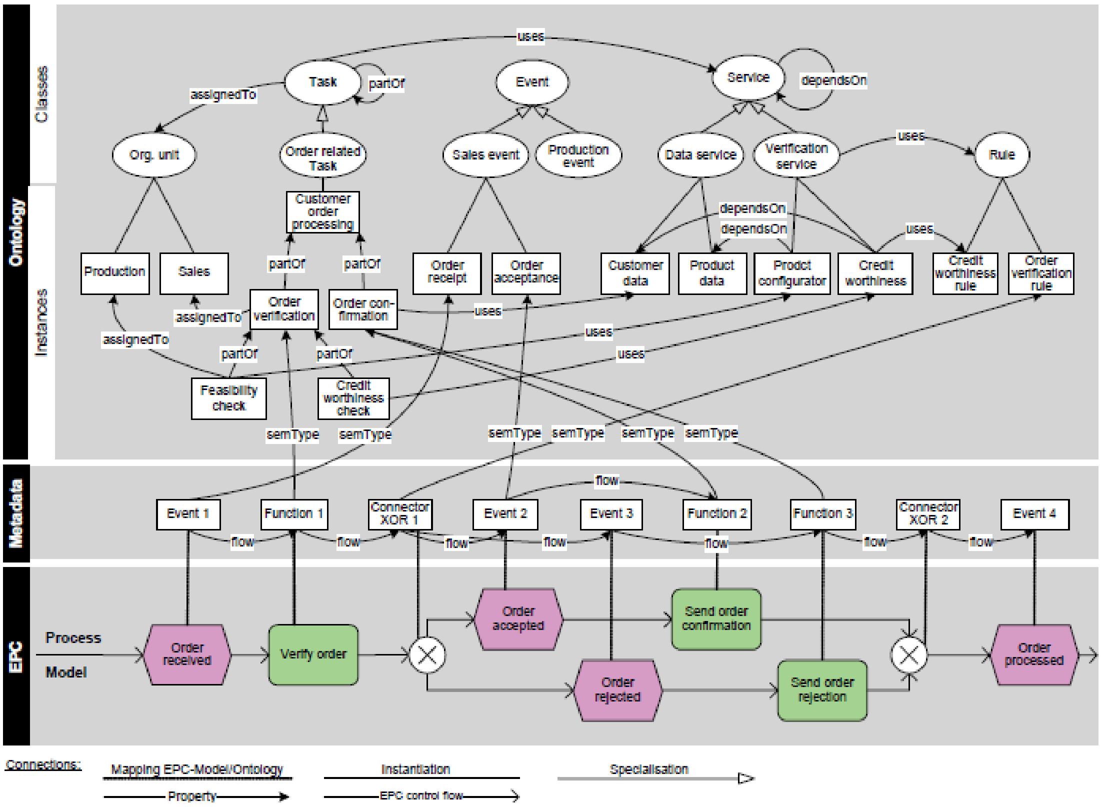


Figure 2.23: Semantically annotated process model "Customer order processing"

They do not refer what metrics are used for the creation of "semType" properties. Their work differs from our approach. We create directly connections between activity labels of business process models and concepts of a domain ontology using three types of similarity measures - string, linguistic and syntactic - as well as filtering. If an accurate domain ontology does not exist a priori, nothing happens until an accurate domain ontology is created.

In [2], Born et al. propose a tool for the user-friendly integration of domain ontology information in the process modeling. They have made extensions to the SAP research modeling tool Maestro for BPMN for the needs of their implementation. In order to match elements of graphical business process model with concepts of domain ontologies, they exploit (1) information about domain objects, actions, states and transitions from the ontology, (2) structural knowledge from the process and (3) a combination of string distance metrics and matching methods considering synonyms and homonyms. Figure 2.24 illustrates the main components of their semantic extension for business process modeling tools. Especially, BPMN data objects and associations are used to describe the activities of a process model more precisely by defining associated objects and their state transitions: Data objects identify the objects an activity deals with and associations link the data objects to the corresponding activities in the process diagram. The user may specify pre- and post conditions for the activities in natural language and may define the objects as well as the objects' states before and after an associated activity has been executed within the graphical model.

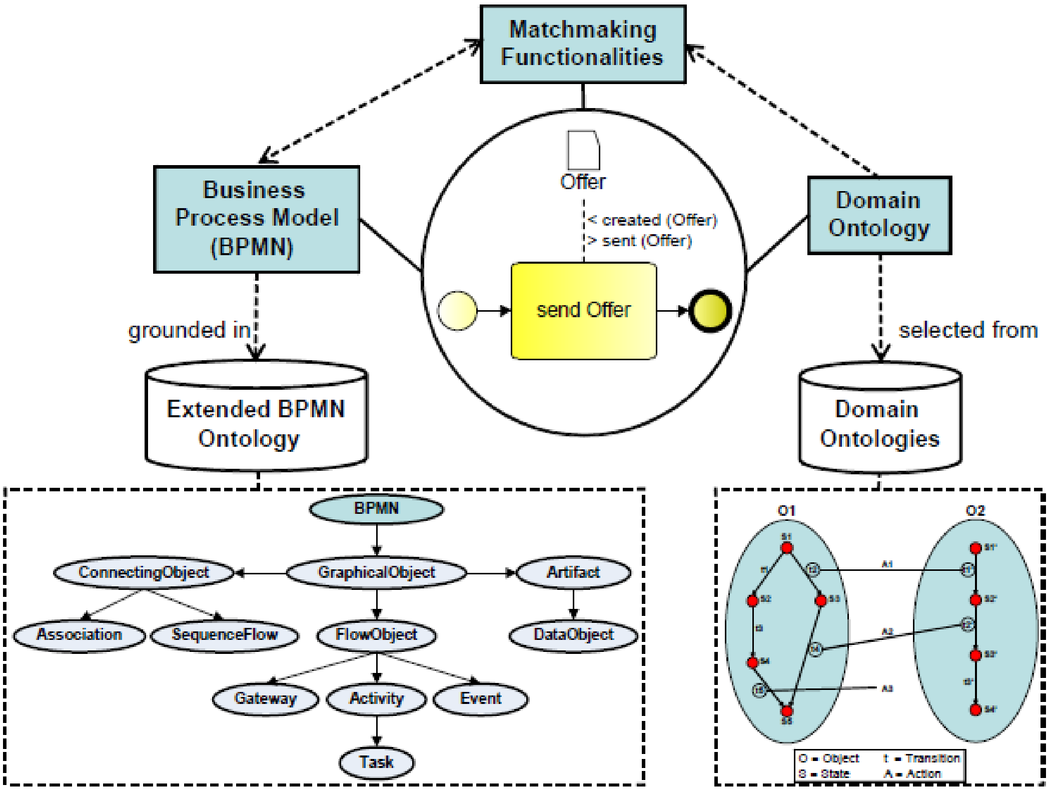


Figure 2.24: Main components of semantic extension for business process modeling tools

Two kind of ontologies are used to achieve the semantic support of modeling activities. First, they have extended the sBPMN ontology of the SUPER project so as to provide possibilities to define states of a data object before and after corresponding activities have been executed, to link objects, states and activities to elements of domain ontologies describing them and to capture natural language pre- and post conditions for activities. Second, they define a possible structure of domain ontologies which cover information concerning domain objects and states which help to model business processes more precisely. Utilizing appropriate domain ontologies, the matchmaking functionalities address the problem of deriving a list of proposals for a selected model element that a user has chosen for semantic annotation. To solve this problem, they propose a combination of string distance metrics and matching methods using synonyms and homonyms, as well as they use process diagram context information and domain ontology knowledge for even better results. They do not refer what exactly string algorithms and matching methods have been used in their approach.

In the SUPER project [8], Dimitrov et al. propose the SUPER ontology stack for the creation of semantic annotations of both BPMN and EPC process models in order to support automated composition, mediation and execution. The core of the SUPER ontology stack is comprised of five ontologies: (1) an Upper Process Ontology (UPO) which defines top-level concepts such as task, condition, etc. (2) a Business Process Modeling Ontology (BPMO) which extends the UPO into a full process ontology providing abstractions over different business process modeling notations such as BPMN and EPC and (3) sBPMN, sEPC and sBPEL which are "ontologised" versions of subsets of the BPMN, EPC and WS-BPEL, respectively. They extend the WSMO studio with a BPMO editor for adding BPMO semantic annotations to existing business process models and for creating new semantic models.

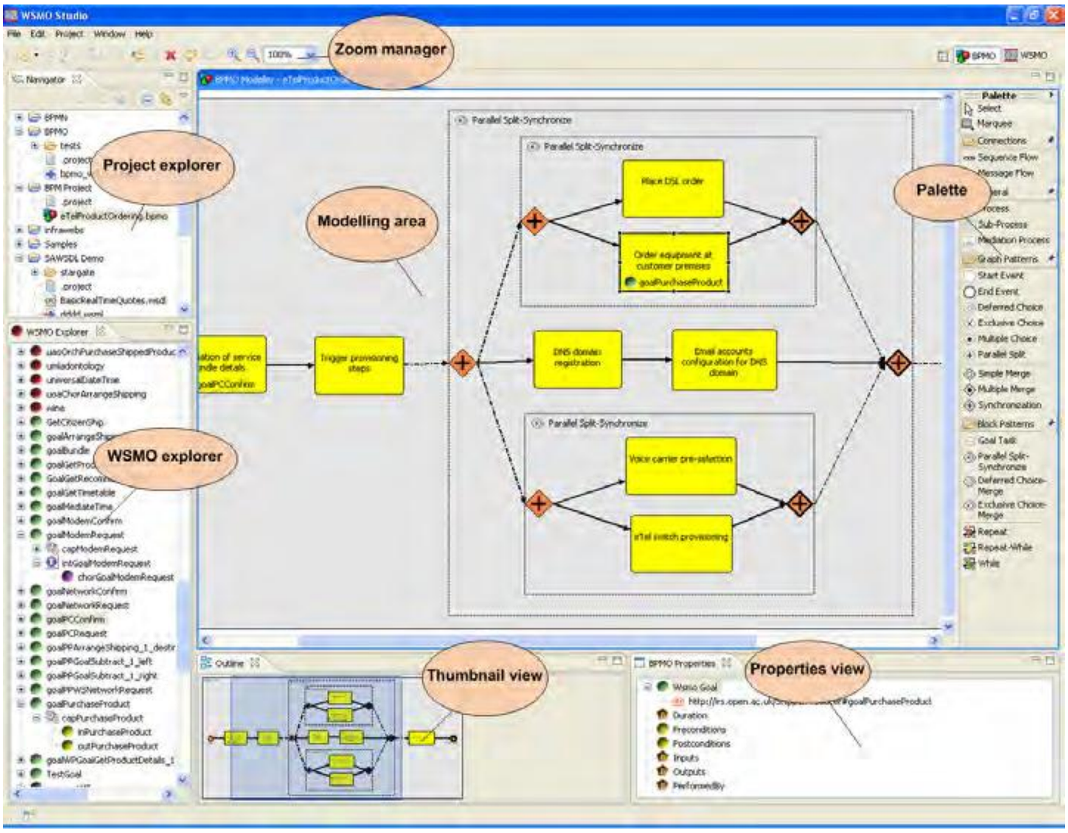


Figure 2.25: WSMO Studio enhancing with a BPMO editor

Figure 2.25 depicts the user interface of WSMO Studio enhancing with BPMO editor. It is based on the BPMN graphical notation extended with BPMO specific modeling primitives and integrated with existing WSMO Studio functionality. The semantically annotated business processes are produced simply by drag & drop existing semantic elements (concepts and instances from reference ontologies) into the relevant element of the process model (activity, data flow elements). They do not refer the formalism and the metrics which are based to produce the semantic annotations for business process models. In addition, WSMO tool supports ontologies expressed in WSML/ WSMO format, whereas we make use of OWL format.

In [36] and [3], Francescomarino et al. propose the representation of semantically labeled business processes as part of a knowledge base that formalizes business process structure, business domains and a set of criteria describing correct semantic annotations on labels. In [36], they provide an ontology integration scheme, where to add semantic annotations to a business process, based on a set of merging axioms that connect BPMN ontology and domain ontology. The merging axioms are in fact a set of constraints which define criteria for correct/ incorrect semantic annotations. An example of such a merging axiom is that a BPMN element of type x can be annotated only with a concept equivalent or more specific than y, where x denotes a concept of BPMN ontology and y denotes a concept of domain ontology. In order to express this kind of constraints and to support automatically verify on them, they propose to encode all the information about semantically annotated processes into a logical knowledge base, called Business Processes Knowledge Base (BPKB), which is composed of the following four modules, as shown in Figure 2.26:

- **BPMN ontology**: formalizes the structure of a Business Process Diagram.
- **Domain Ontology**: is a set of ontologies that describes a specific business domain.
- **Merging axioms**: state the correspondence between the BPMN ontology and the domain ontology.
- **BPD instances**: contain the description of a set of Business Process Diagrams in terms of instances of the BPMN/ domain ontology.

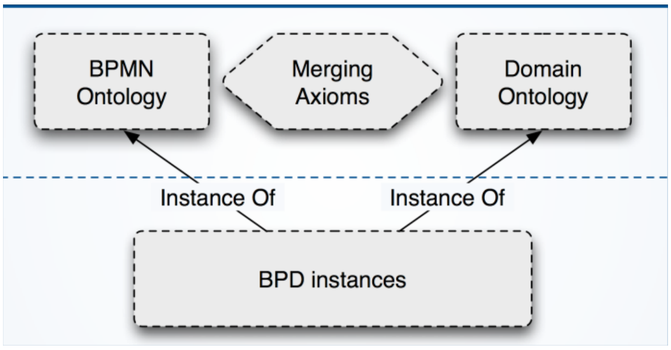


Figure 2.26: Business Processes Knowledge Base

In [3], they extend the structure of Business Processes Knowledge Base, as shown in Figure 2.27, to incorporate constraints used to formalize structural requirements which refer to descriptive properties of the annotated process diagram and not to its execution.

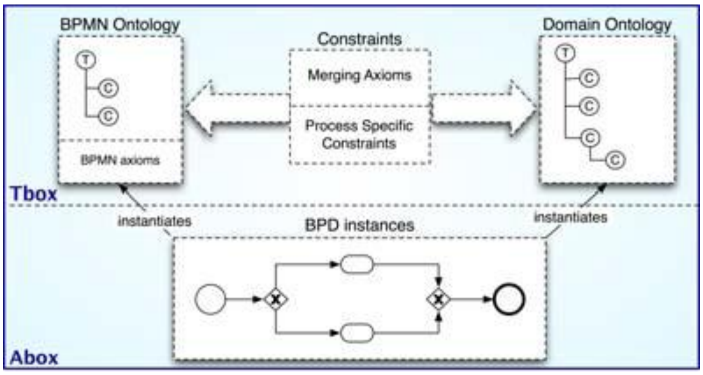


Figure 2.27: Extending Business Processes Knowledge Base

They focus on three types of process specific constraints that can be expressed over the Business Process Diagrams: containment constraints (x contains y), enumeration constraints (x contains at least/ at most/ exactly n objects of type x) and precedence constraints (x is always preceded by y). Finally, they describe a tool for the automated transformation of an annotated business process into an OWL ontology and evaluate how standard DL reasoners can be used to automatically verify these constraints as ontology consistency violations. We make use of a variant of Business Processes Knowledge Base in order to provide process querying mechanisms in our implementation (more details in section 5).

In [7], Ehrig et al. present an approach for (semi-)automatic detection of synonyms and homonyms of process element names in order to support semantic process model interoperability and interconnectivity by measuring the similarity between process models semantically modeled with the Web Ontology Language (OWL). For this purpose, they have developed an OWL-DL based description of Petri nets. In order to achieve their goal, they use three similarity measures: (1) *syntactic* measure, where they compare the number of common characters in the element names using a variant of Levensthein edit distance method, (2) linguistic measure, where the similarity degree based on the WordNet dictionary to determine synonyms sets and (3) structural measure where they consider the context of concept instances and makes it possible to detect homonyms. Finally, they aggregate the three similarity measures to a combined similarity measure as follows: The combined similarity $\mathrm{sim}_{\mathrm{com}}$ between two concept instance names c_1 and c_2, let c_1 be a particular concept instance name of SBPM_1 and c_2 be a concept instance name of SBPM_2, is an aggregation of the degrees returned from the syntactical, linguistic and structural similarity measures having a particular weight each one, w_{syn}, w_{ling} and w_{str}, that can be individually assigned by users or learned for example using machine learning based approaches on a training set, as indicated in the following formula:

$$sim_{com}(c_1, c_2) = \frac{w_{syn}*sim_{syn}(c_1,c_2)+ w_{ling}*sim_{ling}(c_1,c_2)+ w_{str}*sim_{str}(c_1,c_2)}{w_{syn}+ w_{ling}+w_{str}}$$

The similarity between two semantic business process models SBPM_1 vs. SBPM_2 is defined by semantic relationships, which they consider by the two sets of concept instances C_1 and C_2 of SBPM_1 and SBPM_2.

- **equivalence**: $\mathrm{sim}(\mathrm{SBPM}_1, \mathrm{SBPM}_2) = 1$ iff $C_1 = C_2$,
- **disjointness**: $\mathrm{sim}(\mathrm{SBPM}_1, \mathrm{SBPM}_2) = 0$ iff $C_1 \cap C_2 = \emptyset$,
- **intersection**: $\mathrm{sim}(\mathrm{SBPM}_1, \mathrm{SBPM}_2) \in [0...1]$ iff $C_1 \cap C_2 = \{x|(x \in C_1) \wedge (x \in C_2)\} \wedge C_1 \neq C_2$.

Based on these semantic relationships they specify the overall similarity between two semantic business process models as the maximum combined similarity between c_1 and concept instances c_{2j} of SBPM_2, as indicated in the following formula:

$$sim_{SBPM}(C_1, C_2) = \frac{1}{n}\sum_{i=1}^{n}\left(max_{j\in 1m}\left(sim_{com}(c_1, c_2)\right)\right)$$

Very close to our work is the [37]. Ciuciu et al. have developed a tool, called Knowledge Annotator (KA), to provide semantic support to the business modelers during the design of secure business process models. The tool uses ontology-based data matching algorithms and a strategy in order to infer the recommendations the best fitted to the user request, from a dedicated knowledge base and an ontology of security constraints. The goal of ontology-based data matching is to find the similarities between two data sets each of which corresponds to one part of the ontology. The selected strategy is the Controlled Fully Automated Ontology Based Data Matching Strategy (C-FOAM). It is a hybrid strategy which combines (1) string matching algorithms, (2) lexical matching algorithms and (3) at least one graph-based matching algorithm. Specifically, the C-FOAM is based on two modules: (1) the *Interpreter* module and (2) the *Comparator* module, as shown in Figure 2.28.

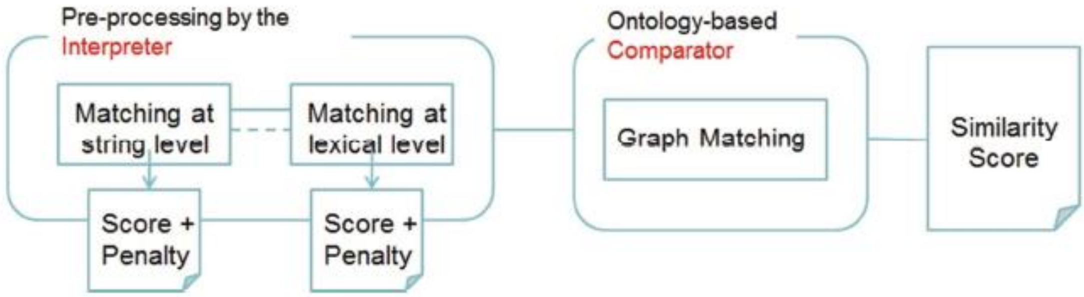


Figure 2.28: C-FOAM model for ontology-based data matching

The *Interpreter*, at first, makes use of the domain ontology and string matching algorithms to interpret the end users' input. In case that the data object, which belongs to the string matching algorithms results, is denoted by several terms, it then makes use of the lexical dictionary, WordNet, to take into account synonyms of them. If a given object term could not be found in the ontology and lexicon, the best fitting data object is returned from the ontology using fuzzy matching based on string similarity. The similarities should be above a certain threshold which is set in the application configuration of their tool. If the data object is found based on fuzzy matching then a penalty percentage will be used on the matching score. Finally, the interpreter will return the correct concept(s) defined in the ontology or lexical dictionary and an annotation set of the concept. The *Comparator* computes the similarity between two found data objects annotated with binary facts from the ontology base. It uses a graph based algorithm or a combination of different graph-based algorithms to find the similarities between the two annotation sets. The *Ontology* which is used by the tool is modeled following the Developing Ontology Grounded Methodology and Applications (DOGMA).

More closely to our work is the [4]. Francescomarino and Tonella have proposed a technique to support the business community with automated suggestions of semantic annotation of process models expressed in BPMN, based on a measure of similarity between ontology concepts and the labels of the process elements to be annotated. Their approach relies on linguistic analysis of the process element labels and of the ontology concepts names. Matching is based on a measure of information content similarity. Specifically, they use a linguistic analyzer, the MINIPAR, to tokenize the process elements labels and the ontology concepts. Moreover, for each word it identifies the grammatical category (e.g. verbs, nouns, adjectives) as well as the grammatical relationship (e.g. verb-object, article-noun, specifier-specified) and the head word guiding such a relationship, if any. They also use the WordNet dictionary to categorize terms according to their meaning (sense) and synonym set (synset). More specific, they use it to solve the semantic ambiguity of ontology concepts by mapping them to unique synsets. The Lin's formula is used to compute the information content similarity between two terms t1 and t2:

$$ics(t1, t2) = \frac{2 * \log\left(p\big(MCSA(t1, t2)\big)\right)}{\log\left(p(t1)\right) + \log\left(p(t2)\right)}$$

where MCSA is the information content of the Most Specific Common Abstraction between the terms t1 and t2. The identified synset will be the one with the best average of maximum information content similarity value computed over all the relative concepts. Once the ontology concepts are linked to a single WordNet sense, the choice of the suggestions is based on the semantic similarity between pairs of BPMN element labels and ontology concepts. The semantic similarity of a pair (l, c) can be based on the semantic similarity between pairs of words respectively in l (W_l = {w_i ∈ Dict|l = w_1 ... w_n}) and in c (W_c = {w_j ∈ Dict| c = w_1 ... w_m}). They define the candidate set of pairs CSP as CSP ⊆ W_l X W_c such that each word w_i∈ W_l and w_j∈ W_c appears at most once in CSP and the total semantic similarity is maximized by CSP. They also give weights to the semantic similarity measures (e.g. the verb has greater importance that the object, in turn more important than the specifier). Once the semantic similarity measure is known for all pairs, they determine the subset of such pairs which maximizes the total semantic similarity. The result is a suggested semantic annotation for each BPMN element. We make use of three similarity measures (string, linguistic, syntactic), as well as we use weighs and filtering techniques for more precise results.

Chapter 3

Basic BPMN Ontology

3.1. A brief Overview

As already has been mentioned, the BPMN language provides powerful graphical representations of business processes that enable human users to model such processes. Nevertheless, its lack of formalized semantics has already led some authors in the development of a BPMN Ontology. In [15], M. Rospocher et al. represent an ontology for the Business Process Modelling Notation which provides a classification of all the elements of BPMN, together with the formal description of the attributes and conditions that state how to combine BPMN elements to form a valid BPD, as they described in BPMN Specification of OMG.

Of course, this is not the only ontology which has been developed for business process modeling. A number of ontologies for business process modeling have already been proposed, the main objectives of which vary to some extent. Others focus on specific business process language, such as in [15] as it has already been mentioned above and in [14] where O. Thomas and M. Fellmann have developed an ontology for semantic EPC. On the other hand, other authors have developed more general ontologies for business processes, such as in [33] where L. Cabral et al. present the Business Process Modeling Ontology (BPMO) which captures domain-independent organizational aspects and control-flow constructs of business notation, process interaction features from BPEL, and service description and invocations for Semantic Web Services (SWS) and in [34, 35] where the General Process Ontology (GPO) is presented. GPO provides a common conceptualization of the concepts used in different process modeling languages.

Due to difficulties to follow the BPMN Ontology, presented in [15] which is the nearest in our investigation, we developed our own basic BPMN Ontology for the facilitating of this research. The basic BPMN Ontology is an ontological formalization of the BPMN specification which encodes the classification of the most basic elements of BPMN, together with the formal representation of the most useful attributes and conditions describing how the elements can be combined to obtain a BPMN process model compliant with the BPMN Specification.

3.2. The Construction of Basic BPMN Ontology

The main purpose of basic BPMN Ontology is to formalize the structural parts of the BPMN language, meaning to describe the most basic elements of BPMN diagrams and how they can be used to compose these diagrams. It is not intended to model the dynamic behavior of a BPMN process, that is how the execution flow proceeds within it. We develop the basic BPMN Ontology in order to exploit it in the context of semantically annotated business processes and as a consequence to support semantically queries on them. For example we would like to be able to retrieve all processes that contains sub-processes about "cart management".

The development of the basic BPMN Ontology was driven and facilitated by the availability of the BPMN Specification as it is described in [20]. In particular, the BPMN Specification is derived in sections each of which describes in details each BPMN element, containing usually the following content:

- a detailed description of the element, together with some general properties and conditions about it.
- a table with the attributes of the element, including its name, its value type and a description about its usage and its special conditions of usage.
- a detailed description about the conditions which must hold for connecting the element with other elements of the BPMN language.
- examples in XML schema describing the element during the execution of process.

The basic BPMN Ontology does not make usage of all the elements, properties, attributes and conditions which are described in BPMN Specification. It formalizes only the most basic of them in order to be used for querying purposes.

The following steps were followed to build the basic BPMN Ontology (they are in correspondence with the steps modelling process as described in [15]).

In the First Step we manually processed the BPMN Specification to identify all the basic elements of the language. We then associated each of these elements to a class in the ontology and at the same time formalized the initial taxonomy of these classes, that is we defined the "is-a" relationship. For example, we defined the class Flow_Objects and that it has three subclasess which are Activity, Gateway and Event.

In the Second Step we filtered the attributes table corresponding to each element, selected the most important and useful for our research and formalized them either as an object property or a datatype property, based on the following general criteria:

(a) The value type of the attribute is another BPMN element: In this case, we formalized the attribute as an object property having as property domain the class associated with the current element and as property range the class corresponding to the element mentioned as value type of the attribute. For example, the attribute SourceRef of the BPMN element Associations has as value type the BPMN elements Artifact and Flow_Objects.

(b) The value type of the attribute is a datatype with no restrictions: In this case, we formalized the attribute as a datatype property having as property domain the class associated with the current element and as property range an OWL datatype compatible to the one specified in the value type of the attribute. For example, the attribute "Text" of the BPMN element "Text Annotation" has as value type "String".

For each attribute, we also formalized its multiplicity details as an OWL cardinality restriction on the class having the attribute. Particularly, (0..1) multiplicity is encoded as "at most one" OWL cardinality restriction, (1..n) multiplicity is encoded as "at least one" OWL cardinality restriction and finally (1) multiplicity is encoded as "exactly one" OWL cardinality restriction.

In the Third Step we added a new attribute for the class of the BPMN element "Activity" and its subclasses "Task and "Sub_Process", namely "Semantic Annotation" and defined it as a datatype property having as property domain the class "Activity" and as property range the value type "anyURI" which is a link to a concept of a domain ontology. For example, as shown in Figure 3.1, on task "choose a product" has been added as semantic annotation the class "to select product" from the domain ontology "onLineShopDomainOntology[1]" which is linkable with this concept of these domain ontology.

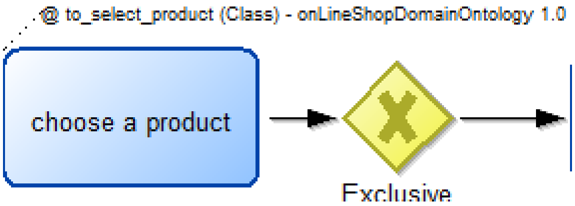


Figure 3.1: Semantic Annotation on task "choose a product"

The basic BPMN Ontology consists of 120 Classes, 36 Object Properties and 15 Data Properties, as shown in table 1. It is not an exhaustive effort of modeling all the attributes, properties and conditions of the BPMN elements as they are presented in BPMN Specification of OMG. We concentrated only to the most basic and useful elements, attributes, properties and conditions which help us to proceed our investigation about semantic annotations on BPMN diagrams. A more detailed development of ontology could be a future work.

[1] The domain ontology "onLineShopDomainOntology", as well as the excerpts of "on-Line Shop" business process have been copied from http://selab.fbk.eu/OnLineShop/

Table 3.1: Basic BPMN Ontology metrics

Basic BPMN Ontology Elements	No
Classes	120
Object Properties	36
Datatype Properties	15

The core component of the basic BPMN Ontology is the Base_Elements Class, divided into seven disjoint sub-classes. Six of them contains the main elements used to describe Business Process Diagrams, which also further divided onto sub-classes. These BPD main element set includes "Flow_Objects" class which is the union of the classes Activity, Gateway and Event, "Connecting_Objects" class which is the union of the classes Sequence_Flows, Message_Flows, Associations and Data_Associations, "Swimlanes" class which is the union of the classes Lane and Pool, "Artifact" class which is the union of the classes Group and Text_Annotation, "Data_Objects" class which is the union of the classes Data_Input, Data_Output and Data_Store and finally "Message" class. The seventh sub-class, namely "Supporting_Elements", contains additional types of elements, mainly used to specify the attribute values of graphical objects. For example, the supporting element "Loop_Characteristics" is used to define the graphical object Activity and signifies that the Activity has looping behavior, i.e. it is repeated sequentially. Except the "Base_Elements" Class, the basic BPMN Ontology contains also the "BPMN_Diagrams" class which contains the basic types of business process diagrams, namely BPMN_process, Choreography and Collaboration. The basic BPMN Ontology does not include elements and attributes which represent Choreography diagrams. A graphical representation of the hierarchy of the classes of the basic BPMN Ontology rooted at Base_Elements class is shown in Figure 3.2.

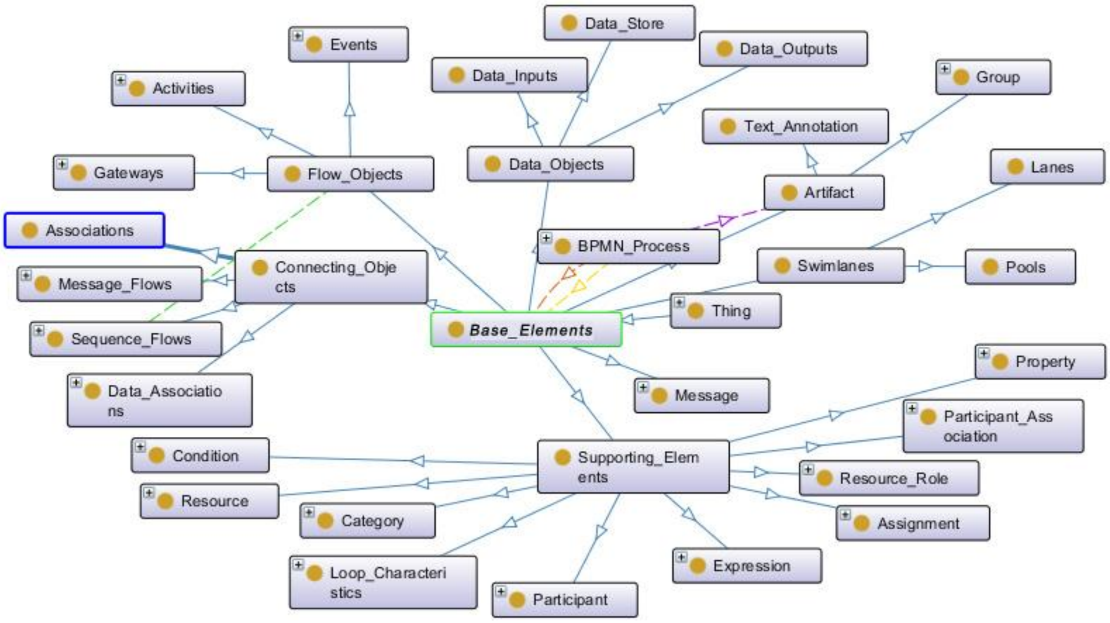


Figure 3.2: An Overview of the "is-a" taxonomy of basic BPMN Ontology rooted at Base_Elements

As far as the Object Properties are concerned, a short description is followed for the most useful of them:

- *has_BaseElements* property and its inverse *isBaseElementsOf* property are used to define the BPMN core elements that are used in graphical representation of a process or sub-process. They have property domain the "BPMN_Process" class and the "Base_Elements" class respectively and property range the "Base_Elements" class and "BPMN_Process" or "Sub_Process" class respectively.
- *has_Artifact* property defines that in a process, sub-process or collaboration may be added as many artifacts as user defines or not. It has property domain "BPMN_Process" or "Sub_Process" or "Collaboration" class and property range the "Artifact" class.
- *has_AssociationSourceRef* property defines the Base Element which the Association is connecting from. It has property domain the "Associations" class and property range the "Artifact" or "Flow_Objects" class.
- *has_AssociationTargetRef* property defines the Base Element that the Association is connecting to. It has property domain the "Associations" class and property range the "Artifact" or "Flow_Objects" class.
- *has_DataAssociationSourceRef* identifies the source of the Data Association. It has property domain the "Data Associations" class and property range the "Data_Objects" class.
- *has_DataAssociationTargetRef* identifies the target of the Data Association. It has property domain the "Data Associations" class and property range the "Flow_Objects" class.
- *has_messageFlow* identifies the two separate pools in a collaboration diagram which the flow of the messages are represented in between two participants which are prepared to send and receive them. It has property domain the "Collaboration Diagram" and property range the "Message_Flows" class.
- *has_messageFrom* property identifies the participant of which sends a message while the *has_messageTo* property identifies the participant on which the message has been sent. They have property domain the class "Message" and property range the class "Participant".
- *has_messageRef* property defines the Message that is passed via the Message Flow. It has property domain the classes "Message_Flows", "Receive_Task" or "Send_Task" and property range the class "Message".
- *has_messageSourceRef* property defines the node that the message flow is connecting from while *has_messageTargetRef* property defines the node that the message flow is connecting to. The first one has property domain the class "Message_Flows" and property range the classes "Activity", "Message_End_Event" or "Message_InterThrow_Event". The other one has property domain the class "Message_Flows" and property range the classes "Activity", "Poll" or "Message_InterCatch_Event".

- *has_sequenceFlow* property identifies the flow objects which participate in a sequence flow (that is the order that the activities will be performed in a process). It has property domain the class "Flow_Objects" and property range the class "Sequence_Flows".
- *has_sequenceSourceRef* property defines the node that the sequence flow is connecting from while the *has_sequenceTargetRef* property defines the node that the sequence flow is connecting to. The first one has property domain the class "Sequence_Flows" and property range the classes "Activity", "Gateway", "Start_Event", "InterCatch_Event", "InterThrow_Event", "NonInterrupt_Intermediate_Event" or "NonInterrupt_Start_Event". The other one has property domain the class "Sequence_Flows" and property range the classes "Activity", "Gateway", "End_Event", "InterCatch_Event", "InterThrow_Event", or "NonInterrupt_Intermediate_Event".

Finally, as far as Datatype Properties are concerned, the most significant for our investigation is the property has_SemanticAnnotation which identifies the semantic annotation that has been added in an activity of a process. It has property domain the class "Activity" and property range the value type "anyURI" which is a link to a concept of a domain ontology.

We could also represent the semantically annotated processes as part of a knowledge base (KB), in correspondence with the [36], which is composed of the following three modules:

(1) The **basic BPMN Ontology** which is a general ontology and provides a formalization of the structural part of business process diagrams, meaning that it describes the most basic BPMN elements and how they can be connected for the construction of BPDs.

(2) The **Domain Ontology** which consists of a set of ontologies that describes a specific business domain.

(3) The **BPD Instances** which contain the description of a set of semantically annotated BPDs in terms of instances of the basic BPMN ontology and the domain ontology. Every element of the semantically annotated business process is represented as an individual of a class.

The first two modules constitute the terminological part, that is the TBox, which represents the background knowledge and the knowledge relevant for the described domain and the last one constitutes the changeable part, that is the ABox, which contains knowledge about the individuals of a specific business process description, as shown in the Figure 3.3.

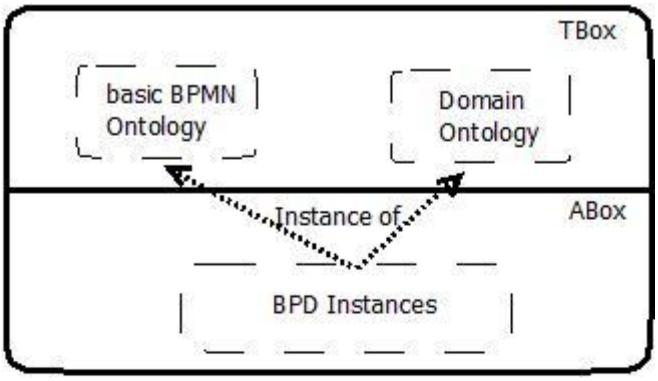


Figure 3.3: Knowledge Base of Business Processes

The reason of encoding a BPD as a set of instances of the basic BPMN Ontology is that reasoning services can be then implemented on it. One of them, that is represented on this work, is the querying on the BPD instances. Specifically, given an instantiated basic BPMN ontology, we provide process querying mechanisms that exploits the information formalized in the basic BPMN ontology. We used SPARQL for encoding the queries which run on the instantiated BPMN ontology.

Chapter 4

Presentation of the Process Following for Semi-Automated Semantic Annotations to Business Processes

In this chapter, we provide a more detailed presentation of similarity measures following to annotate business process models with semantically semi-automated suggestions derived from domain ontology (ies). To help business designers annotate their process models, different matchmaking functionalities are used to link the model elements and specifically activity labels (task and sub-process) of them to available domain concepts. The similarity metrics that we exploit in order to support business analysts with semantic annotation suggestions semi-automatically generated are: *String metrics, linguistic metrics* and *syntactic metrics*. The linguistic and syntactic metrics are based on one of the most known English dictionaries, the WordNet.

WordNet is a large lexical database of English which was developed at Princeton University. It contains nouns, verbs, adjectives and adverbs which are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The majority of the WordNet’s relations connect words from the same part of speech (POS). Thus, the WordNet consists of four sub-nets, one each for nouns, verbs, adjectives and adverbs. In particular:

The main relation among words in WordNet is **synonymy**, that is words that denote the same concept and are interchangeable in many contexts. Synonymy are grouped into unordered sets, called synsets. Each synset contains a brief definition, called gloss and, in most cases, one or more short sentences illustrating the use of the synset members. Words with several distinct meanings are represented in as many distinct synsets. Thus, each form-meaning pair in WordNet is unique.

Furthermore, WordNet contains encoded relation among synsets called **IS-A** relation, **hyperonymy** or **hyponymy**. It links more general synsets to increasingly specific ones. For example, the synset "cutlery" is linked with "spoon" and "teaspoon" meaning that the category cutlery includes spoon, which in turn includes teaspoon and conversely concepts like spoon and teaspoon make up the category cutlery. All the noun hierarchies end up to the root node "entity". Hyponymy relation is also transitive. For example, if a teaspoon is a kind of spoon and a spoon is a kind of cutlery, then a teaspoon is also a kind of cutlery.

WordNet contains also a part-whole relation which is called **Meronymy** and holds between synsets like "seat" and "legs". Parts are inherited from their superordinates. For example, if a chair has legs, then an armchair has legs as well. Parts are not inherited "upward" meaning that if chairs and kinds of chairs have legs, it does not mean that all kinds of furniture have legs.

Except from nouns, verb synsets are organized into "IS-A" hierarchies as well. Verbs towards the bottom of the trees (troponyms) express increasingly specific manners characterizing an event, as in "communicate-talk-whisper". In contrast, Adjectives are organized in terms of **antonymy**. Pairs of "direct" antonyms like wet-dry and young-old reflect the strong semantic contract of their members. Each of these polar adjectives in turn is linked to a number of "semantically similar" ones like dry is linked to parched, arid, dessicated and bone-dry and wet to soggy, waterlogged, etc. As for adverbs, the WordNet contains only few of them (hardly, really, etc.), as the majority of English adverbs are straightforwardly derived from adjectives via morphological affixation (surprisingly, strangely, etc.)

For our implementation we used the eclipse IDE [55] and two java libraries for WordNet. The jwnl 1.4-rc3 java library[38] [39] [40] is an API which was developed by John Didion at Stanford University for not only accessing WordNet data in a programmatic way but also for finding relationships of a given type between two words. The jwi 2.3.3 [41] is a smaller java library which has been developed by MIT for WordNet accessing. We used it in order to get the parts of speech of each word and to compare only words of the same part of speech, e.g. verbs-verbs, nouns-nouns, since the reduction of the comparisons would allow us to limit the time at our disposal and the memory usage. Finally, we exploit only the "synonymy" and the "is-a" relationships of WordNet both for nouns and verbs.

In the following subsections we describe the above similarity metrics in more details, as well as the whole procedure which was followed to find similarities between BPMN process labels and domain ontology concepts.

4.1. String Matching Algorithms

Various string matching algorithms are used to calculate the similarity between two words or short sentences based on the string similarity of their description. In this work, we used character based metrics, token based metrics and phonetic similarity metrics. In particular, the following string algorithms are used:

N-gram algorithm [42]: N-gram is a contiguous sequence of n items (e.g. letters, words) from a given sequence of text. In fact, N-grams are overlapping substrings. The sequences of the N-grams are saved in a string array of N-grams. A n-gram of size 1 is called "unigram", size 2 is called "bigram" or "digram", size 3 is called "trigram". Larger sizes are referred to by the value of n, e.g. "four-gram" and so on. In our research we used the token based similarity metrics: "Bigram or Digram Algorithm" and "Trigram Algorithm". According to **Bigram** or **Digram Algorithm** [43], strings are compared as sequences of two adjacent characters. It is a n-gram with n=2. For example, if we want to find the similarity between the two short sentences: "select product" and "choose product", the algorithm will generate two string arrays with the following sequences: [{se}, {el}, {le}, {ec}, {ct}, {t }, { p}, {pr}, {ro}, {od}, {du}, {uc}, {ct}] and [{ch}, {ho}, {oo}, {os}, {se}, {e }, { p}, {pr}, {ro}, {od}, {du}, {uc}, {ct}]] respectively and will compare the corresponding sequences. Thereafter the number of common sequences are used to Dice's coefficient statistic formula [44] for comparing the similarity between the two sentences, as shown in (1).

$$QS = \frac{2|X \cap Y|}{|X| + |Y|} \qquad (1)$$

Where the |X| and |Y| are the number of sequences in the two short sentences. The QS is the quotient of similarity which ragnes from 0 to 1.

In correspondence, in the **Trigram Aglorithm** [45], which is a special case of the n-gram where n=3, strings are compared as sequences of three adjacent characters with the same way as described in the Bigram Algorithm.

Levenshtein Distance Algorithm [46] [47]: It is a string metric for measuring the difference between two sequences. In fact, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to transform one word into the other. Mathematically, the Levenshtein distance algorithm between two strings s1 and s2 with lenghts |s1| and |s2| respectively is given by the formula, as shown in (2):

$$lev_{s1,s2}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} lev_{s1,s2}(i-1,j) + 1 \\ lev_{s1,s2}(i,j-1) + 1 \\ lev_{s1,s2}(i-1,j-1) + 1_{(s1_i \neq s2_j)} \end{cases} & otherwise \end{cases} \qquad (2)$$

where the $lev_{s1,s2}(i,j)$ is the distance between the first **i** characters of **s1** string and the first **j** characters of string **s2**. The $1_{(s1_i \neq s2_j)}$ is equal to 0 when $s1_i = s2_j$ and equal to 1 otherwise. Furthermore, the first element of the minimum corresponds to deletion, from s1 to s2, the second element corresponds to insertion and the last one to match or mismatch, depending on whether the respective symbols are the same. For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other.

1. kitten to sitten : substitution of "s" for "k".
2. sitten to sittin : substitution of "i" for "e".
3. sittin to sitting : insertion of "g" at the end.

Jaro - Winkler Distance Algorithm [48] [49]: It is a string metric for measuring the similarity between two, especially short, sentences. It is based on the number and order of the common characters between two strings. Mathematically, the jaro distance $\mathbf{d_j}$ between two strings **s1** and **s2** is given by the formula, as shown in (3):

$$d_j = \begin{cases} 0 & if\ c = 0 \\ \frac{1}{3}\left(\frac{c}{|s1|} + \frac{c}{|s2|} + \frac{c-t}{c}\right) & otherwise \end{cases} \qquad (3)$$

where **c** is the number of common characters and **t** is half the number of transpositions. Two characters of the strings **s1** and **s2** respectively, are considered common only if they are the same and not farther than *floor(max(|s1|,|s2|)/2).* Each character of **s1** is compared with all its matching characters in **s2**. As number of transpositions (**t**), it is defined the number of common characters but in different sequence order divided by 2. For example, if we want to compare the string "crate" with the string "trace", they have "r", "a" and "e" as common characters, i.e. c=3. Although the letters "c" and "t" appear in both strings they are farther than 1, as floor(5/2)>1, so transpositions t=0. Winkler added on jaro distance algorithm a prefix scale **p** which gives more favorable ratings to strings that match for the beginning for a set prefix length **l**. Therefore, given two strings **s1** and **s2**, their jaro - winkler distance $\mathbf{d_w}$, as shown in formula (4), is:

$$d_w = d_j + (lp(1 - d_j)) \qquad (4)$$

where $\mathbf{d_j}$ is the jaro distance between the two strings **s1** and **s2**. **l** is the length of common prefix at the start of the string up to a maximum of 4 characters, as it has been defined by Winkler. **p** is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. The standard value for this constant in Winkler's work is **p = 1/10**. The higher the jaro-winkler distance for two strings is, the more similar the strings are. The score is normalized in such a manner that 0 equates to no similarity while 1 is an exact match. For example, if we want to compare the string **s1** "martha" and the string **s2** "marhta", we find that they have 6 common characters, i.e. c=6, the length of string s1 is |s1|=6, the length of string s2 is |s2|=6, there are mismatched characters "th" of string s1 and "ht" of string s2 leading to **t** =2/2 = 1. So, the jaro score is:

$$d_j = \frac{1}{3}\left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944$$

adding the Winkler's standard weight **p=1/10** and finding that the length of common prefix between the two strings is **l=3**, the jaro - winkler score is:

$$d_w = 0.944 + \left(3 * 0.1 * (1 - 0.944)\right) = 0.961$$

Jaccard Similarity Algorithm [50] [48]: It is a token based metrics which is used to measure the similarity between two finite sets. It is defined as the size of the intersection divided by the size of the union of the two sets. Given two strings **S** and **T** respectively, the formula which gives the Jaccard Coefficient **j(S, T)**, as shown in (5), is:

$$j(S,T) = \frac{|S \cap T|}{|S \cup T|} \qquad (5)$$

where the intersection of the two strings, denoted as $|\boldsymbol{S} \cap \boldsymbol{T}|$, gives the number of common characters in both strings while the union of the two strings, denoted as $|\boldsymbol{S} \cup \boldsymbol{T}|$, gives all the characters which are in either string. The jaccard similarity coefficient ranges from 0 to 1. For example, the similarity between the strings "kitten" and "sitting" is:

$$jaccard = \frac{4}{9} = 0.44$$

while the jaccard coefficient for the strings "apple" and "apple pie" is:

$$jaccard = \frac{5}{9} = 0.56$$

Char Frequency Similarity Algorithm [53]: It is a string metric for quickly estimating how similar two strings are. Specifically, it searches the occurrences of characters in two strings and computes the similarities based on the character occurrence list. For example, the char frequency similarity between the words "select" and "action" is 0.334.

Soundex Algorithm [51] [52]: It is a phonetic algorithm for indexing names or general words by sound, as pronounced in English. The goal is for homophones (pronounced the same as another word but differs in meaning, and may differ in spelling) to be encoded to the same representation so that they can be matched despite minor differences in spelling e.g. beer - bear. It is especially useful in the case where words can be misspelled or have multiple spellings where soundex algorithm can find similar sounding terms. The algorithm mainly encodes consonants. A vowel is encoded only if it is the first letter. The main principle used by Soundex is based on the six phonetic classifications of human speech sounds (bilabial, labiodental, dental, alveolar, velar, and glottal). Each consonant is grouped in one of the above six categories depending on where you put your lips and tongue to make the sound of each one. It aims to find a code for every word. This code consists of a letter followed of three numerical digits. The letter is the first letter of the word and the digits encode the remaining consonants of the word. The steps to find the soundex code is as follow:

1. Retain the first letter of the word and drop all other occurrences of a, e, i, o, u, y, h, w.
2. Replace consonants with digits as follows (after the first letter):
   1 = b,f,p,v.
   2 = c,g,j,k,q,s,x,z.
   3 = d,t.
   4 = l.
   5 = m,n.
   6 = r.
3. If two or more letters with the same number are adjacent in the original name (before step 1), only retain the first letter. Also two letters with the same number separated by 'h' or 'w' are coded as a single number, whereas such letters separated by a vowel are coded twice. This rule also applies to the first letter.
4. If you have too few letters in your word that you cannot assign three numbers, append with zeros until there are three numbers. If you have more than 3 letters, just retain the first 3 numbers.

For example, the words "Ashcroft" and "Asicroft" are encoded in A261 and A226 respectively. In the first word the chars "s" and "c" would receive a single number of 2 and not 22 since an "h" lies in between them, whereas in the second one the chars "s" and "c" would receive a double number of 2 (22) since a vowel "i" lies in between them. In the case of words "Ashcroft" and "Ashcraft" both yield A261 for the same reasons as described above. In the first example the soundex returns 0 (zero) score whereas in the second one it returns 1.

Matching at string level is easy to implement and does not require a complex knowledge resource. A natural language description of the objects/or concepts and/or its composing elements is sufficient [37].

4.2. Linguistic Metrics

More complex but also more accurate metrics [54]. They calculate the similarity of two words or short sentences based on the semantic similarity of their descriptions. They rely on a dictionary to determine synonyms. The dictionary which is incorporated in our work is the WordNet. As it has already underlined above, it [26] is one of the most known English language thesaurus allowing to categorize terms according to their meaning and synonym set, called synset.

In detail, given two short sentences, for example an activity label and a domain ontology concept, we tokenize them, meaning that we identify the words and the punctuation symbols which constitute the sentence, and extract two list of words respectively. From these lists, we have removed unnecessary common tokens like "the", "a", "an", "of", "and", "for" and "to" because they do not have any sense in the comparison procedure. Then, for each word, we identify the grammatical category in which it belongs. Specifically, the set of the classification of grammatical categories is {verb, noun, adjective, adverb} following the four categories of WordNet. With this way, we compare only the terms of each list of the same grammatical category, that is verb-verb, noun-noun, etc., lessening the unnecessary comparisons. Finally, we find the synonym terms using the corresponding relation of the WordNet and the similarity score of each similarity pair which range from 0 to 1. For example, we find that the short sentences: "choose a product" and "to_select_product" are synonyms with similarity score at 1.

4.3. Syntactic Metrics

In order to further improve the matching procedure, we also exploit the "is-a" relationship of the WordNet dictionary. Hypernyms or hyponyms (which constitute the "is-a" taxonomy) are used to take into account more generic or more specific terms of a given term. For example, if we search for semantic annotation into the task label "choose a product", the system will retrieve not only the synonym term "to_select_product" from the selected domain ontology but also the more generic and more specific terms from the same selected domain ontology, ranked according to the calculated similarity score which range from 0 to 1 in descending order (how the similarity score is calculated described in the next sub-section): {"to_select_product_quntity", "to_select_product_category", "product", "product_quantity, "product_category", "product_data", "product_availability", "to select_quantity", "to_select_cateogory"}.

4.4. Combined Similarity Measures

In order to suggest the most relevant semantic annotations to business analysts, the system must compute a combined similarity degree between a BPMN activity label and concepts of a selected domain ontology. The combined similarity degree is an aggregation of the degrees returned from the string, linguistic and syntactic metrics as explained above. For aggregating the results, parameters can be set for weighting the strength of each string matching algorithm. In addition, for computing the final aggregated result, a filtering method of the results has been implemented in order to accurately and efficiently detect the semantic annotations for BPMN activity labels. The whole matching process consists of eight phases as shown in Figure 4.1: (a) Loading, (b) String Matching, (c) Weighting String Algorithms, (d) Linguistic Matching, (e) Syntactic Matching, (f) Mappings, (g) Filtering (threshold), (h) Outputs.

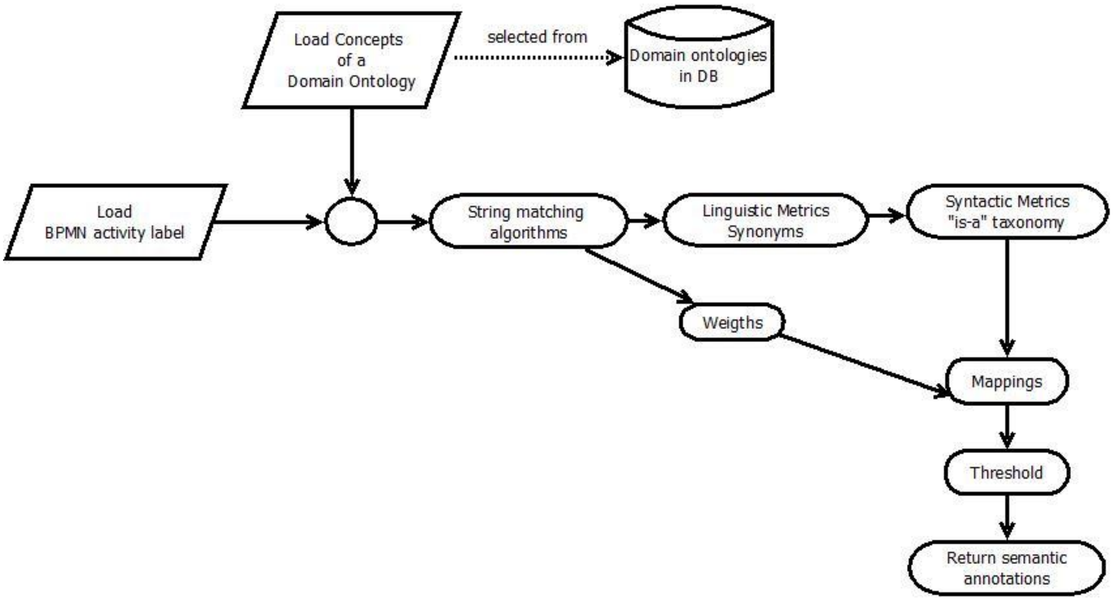

Load Concepts of a Domain Ontology
selected from
Domain ontologies in DB
Load BPMN activity label
String matching algorithms
Linguistic Metrics Synonyms
Syntactic Metrics "is-a" taxonomy
Weigths
Mappings
Threshold
Return semantic annotations

Figure 4.1: Flow Chart of Matching Process

At this point, it is important to note that the end user of the matching process is a business analysts who may possibly lack a sufficient background knowledge of ontologies and matching algorithms. For this reason, we do not give him the permission to customize parameters like selection string algorithms, weights and thresholds for the matching process. Because of his lack of knowledge, the opposite course of action would lead to insufficient customization followed by insufficient results.

The first phase of the matching process is the loading of a BPMN activity label and all concepts of a selected domain ontology. When the user types an activity label, the system (SeMFIS) searches for the most accurate domain ontologies of its database (how it is processed by the system so as to purpose accurate ontologies is described in a following section). If there is one, the user selects it and the system loads the activity label and all concepts of the selected domain ontology to the matching process. If not, nothing is done until a suitable domain ontology is inserted by the protégé via SeMFIS plugin in xml format or it is created in the system's database in owl format or when an existing domain ontology is updated with suitable classes.

The second and third phase of the matching process is the implementation of all the string matching algorithms which has already specified above (bi-gram, tri-gram, Levenstein, Jaro-Winkler, Jaccard, char frequency, soundex). However, all the aforementioned algorithms do not have the same weight in the matching process. Thus, bi-gram, tri-gram and soundex algorithm have fixed weight 0.5, Levenshtein distance algorithm has fixed weight 0.7 and Jaro-Winkler, Jaccard and char frequency algorithm have fixed weight 0.8. It should be noted at this point that the fixed weights of the available matching algorithms are indicative. They are customized after testing and by using our own experience. Of course, they can be customized again in the future. Each one of the above algorithms gives a score of similarity ranged from 0.0 to 1.0.

The fourth and fifth phase of matching process are the implementation of linguistic and syntactic metrics using the "synonyms" and "is-a" relation functions of the English dictionary WordNet. The weight of these two metrics can be considered 1.0. The score of similarity ranged from 0.0 to 1.0.

After applying all matching algorithms, in the next phase of matching process, a weighted average of similarities is calculated with the following formula (6):

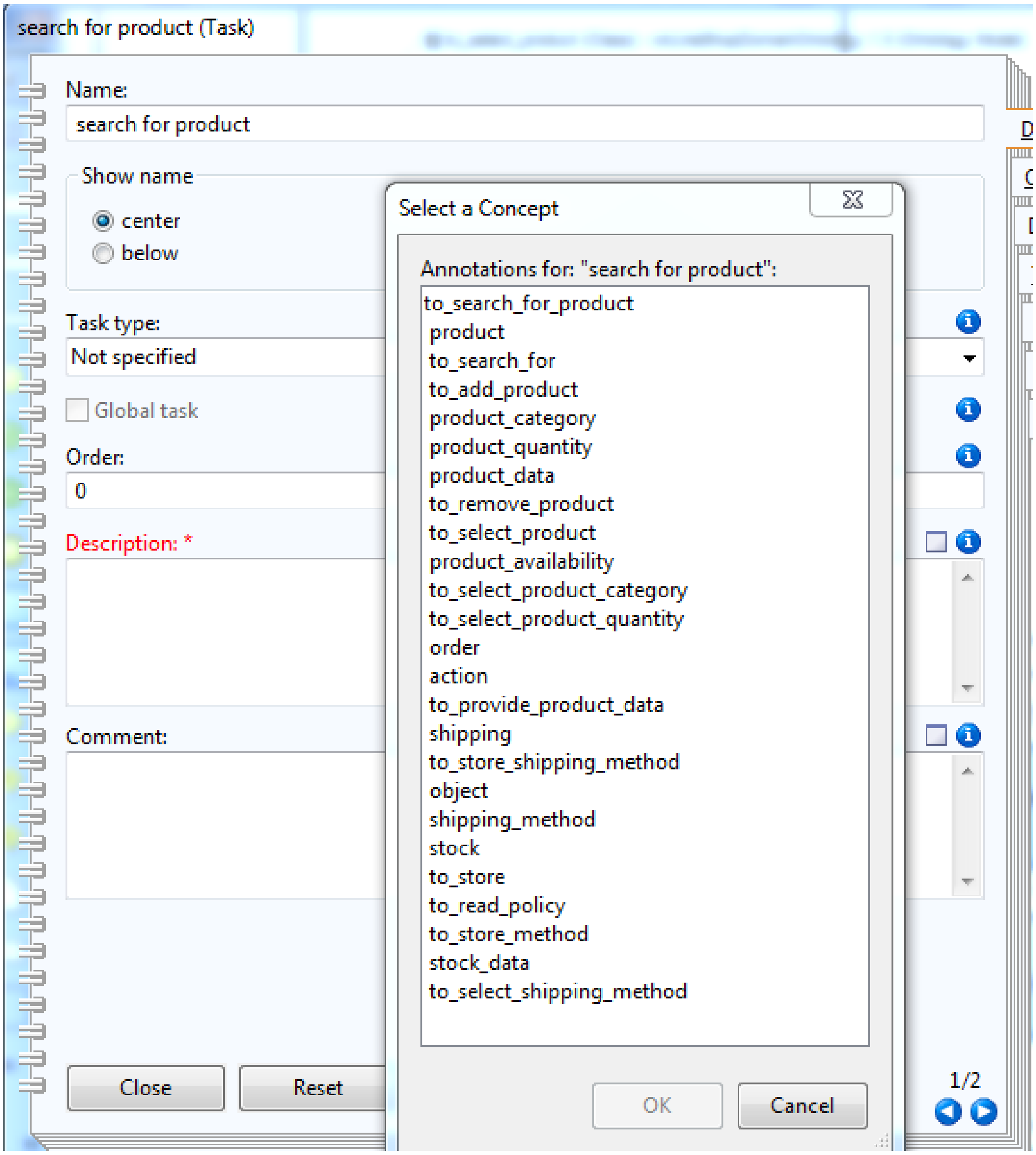
$$sim = \frac{\sum_{n=0}^{i} sim_{i} * w_{i}}{\sum_{n=0}^{i} w_{i}} \qquad (6)$$

where **sim** = {sim_1,sim_2, ... , sim_i} are the calculated value similarities from all matching algorithms and **W** = {w_1, w_2, ... , w_i} are the corresponding weights of each matching algorithm which are fixed from the system whereas **n** is the number of similarity measures which are used in our implementation.

The same process is followed for all the matching annotations for each matching pair between the loaded activity label and each of the loaded concepts of the selected domain ontology and the similarity scores are stored in a similarity matrix.

In the next step of matching process, a fixed threshold has been defined by the system in 0.5. The fixed threshold is indicative, and so are the fixed weights of the available matching algorithms as stated earlier. It is customized after testing and, of course, it can be re-customized in the future.

When there are no more matching pairs to compare, the system checks the calculated similarity score of each matching pair and if it is above the given threshold, then the domain ontology concept of this pair is considered an accurate match and might be added into the result map.



search for product (Task)

Name: search for product

Show name: ● center ○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

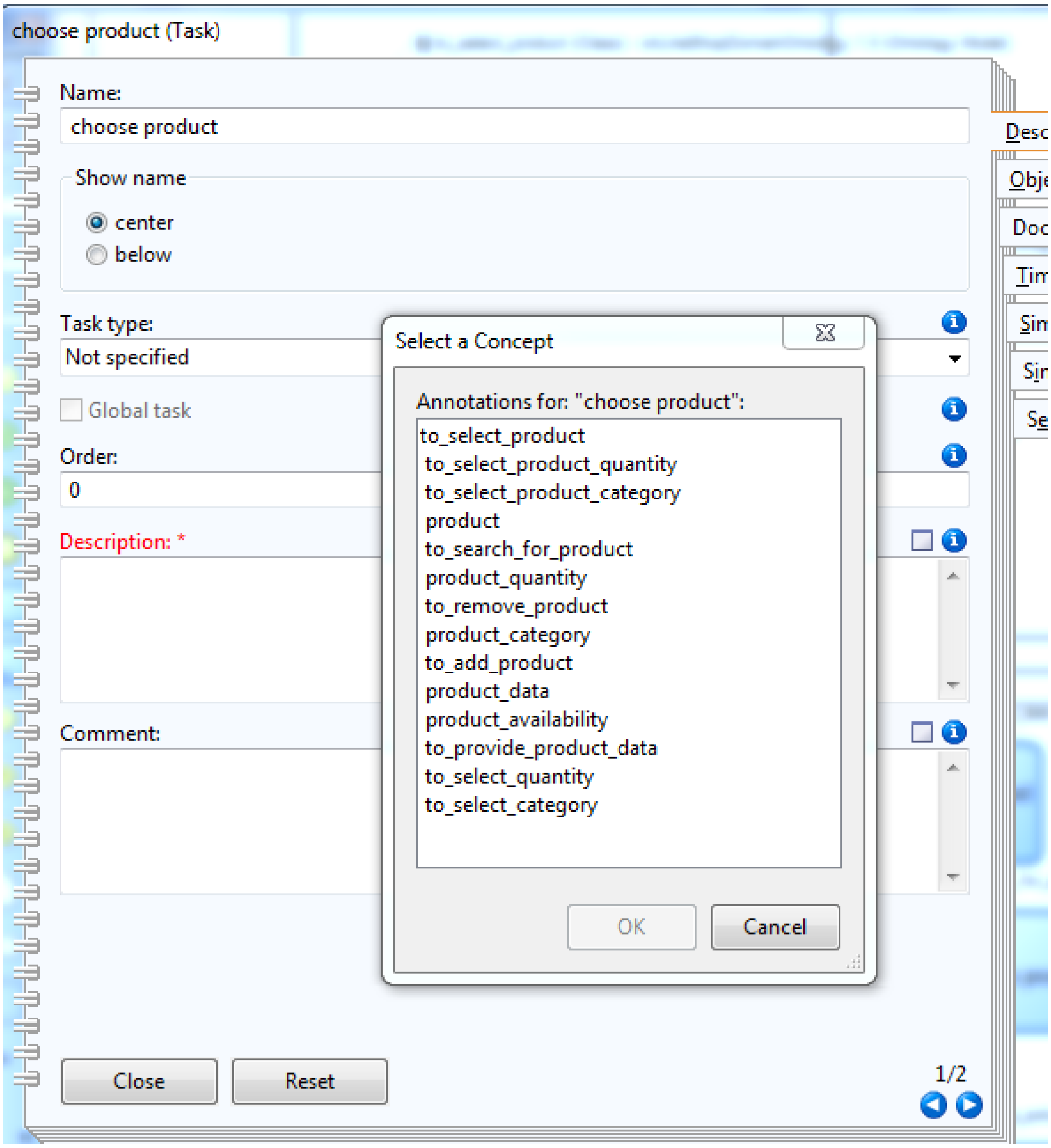
Select a Concept

Annotations for: "search for product":

to_search_for_product
product
to_search_for
to_add_product
product_category
product_quantity
product_data
to_remove_product
to_select_product
product_availability
to_select_product_category
to_select_product_quantity
order
action
to_provide_product_data
shipping
to_store_shipping_method
object
shipping_method
stock
to_store
to_read_policy
to_store_method
stock_data
to_select_shipping_method

Figure 4.2: Semantic annotations list for the BPMN Task label "search for product"

In the final phase of matching process, a list of the semantic annotations which are above the 0.5 threshold are presented to the user. Thus, every loaded BPMN activity label in the matching process is allowed to match with more than one loaded domain ontology concept (1:n). That of course is based on the fact that we search not only for the exactly synonym concepts of the loaded activity label but also for parents and children of it. The list is ranked from the most accurate semantic annotation to less accurate one (that is annotations with similarity score from 1.0 to 0.5) as shown in the Figure 4.2. As we can see from the Figure 4.2, the top of the list is occupied by suggested annotations derived from string matching algorithms such as {to_search_for_product, product, to_search_for, to_add_product, product_category, product_quantity, etc} which have higher scores of our method and following suggested annotations derived from syntactic metrics ("is-a" relationship) such as {order, action, object, stock, to_store, to_read_policy, stock_data, etc} with a score of up to 0.5.



choose product (Task)

Name: choose product

Show name
● center
○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

Select a Concept

Annotations for: "choose product":

to_select_product
to_select_product_quantity
to_select_product_category
product
to_search_for_product
product_quantity
to_remove_product
product_category
to_add_product
product_data
product_availability
to_provide_product_data
to_select_quantity
to_select_category

Figure 4.3: Semantic annotations list for the BPMN Task label "choose_product"

In the Figure 4.3, we can see another example of automatically suggested annotations where have been involved all three metrics of our method (string, linguistic, syntactic). Specifically, in this Figure we want to match an annotation to the task label "choose product". The system suggests a list of ranked annotations from the most accurate annotation to the less one. The synonym sentence {to_select_product} is derived from linguistic metrics, as the verb "select" is synonym with the verb "choose" and it is in the top of the list as it has the highest score of all the concepts. Following concepts which are derived from linguistic and syntactic metrics such as {to_select_product_quantity, to_select_product_category} which are "children" of the sentence "to_select_product", as well as concepts which are substrings of the label name such as {product, product_data}.

The user can select only one suggested annotation from the list or none choosing the "cancel" button. If he/ she chooses the "cancel" button, the notebook of the specific task opens into the "semantic annotation" chapter and the user could select manually a domain ontology and then a concept which he/ she considers most accurate for the specific task, as shown in the Figure 4.4. In this Figure, the "semantic annotation" chapter of the task "read policy" has automatically opened and the user matches manually the annotation "to_read_policy" (blue rounded rectangle) from the ontology "onLineShopDomainOntology" (red rounded rectangle) in the task label "read policy".

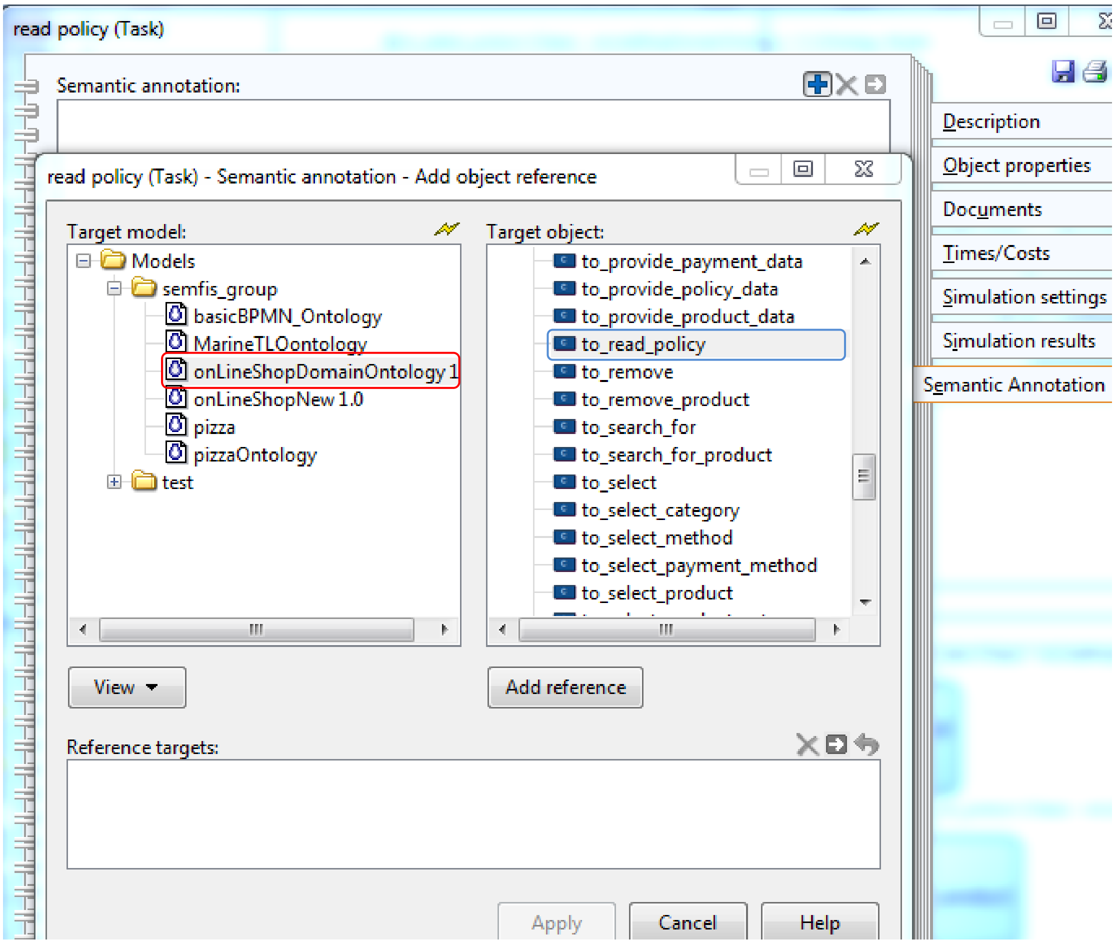


Figure 4.4: Manually selection of semantic annotation

4.5. Suggesting accurate domain ontologies and annotated BPMN processes

As shown in Figure 4.1, the SeMFIS may have a lot of domain ontologies in its database which have either been created immediately in the system by ontology experts or have been inserted in it from the protégé program via the SeMFIS plugin. Before the system puts forward automatic semantic annotations, it searches for the most accurate domain ontologies in its database where the semantic annotations could be extracted from, and suggests them to the user in order to select one. (The whole procedure on how the system searches for the most accurate domain ontologies is described with details below). If an appropriate domain ontology does not exist in its database, the system does not make any automated suggestion. It also discloses already annotated BPMN models which have been annotated with domain ontology concepts related with the activated activity label and suggests them too. If the user selects an already annotated BPMN model the system saves time as it does not need to search in all concepts of a domain ontology but only between the concepts which have already been annotated in the selected annotated BPMN model.

The whole procedure of detecting the most accurate domain ontologies of the database, as well as the most accurate annotated BPMN models, if they exist, is approximately the same as the automatic detection of semantic annotations. We use the same similarities measures, that is all the string algorithms, linguistic and syntactic metrics, as well as their fixed corresponding weights, as they have already been underlined earlier. The only difference is that we implement a two phase filtering method in the final similarity score, instead of one, in order to detect the most accurate domain ontologies and annotated BPMN models. The Figure 4.5 shows the whole procedure.

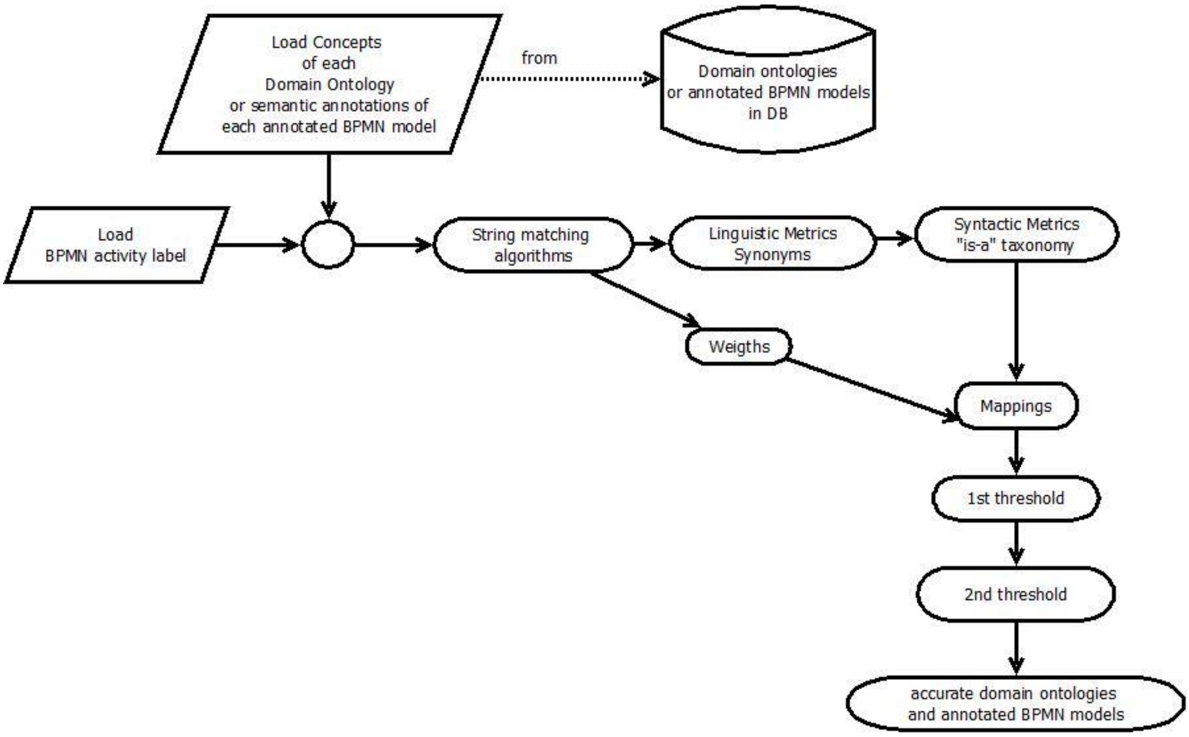

Load Concepts of each Domain Ontology or semantic annotations of each annotated BPMN model
from
Domain ontologies or annotated BPMN models in DB
Load BPMN activity label
String matching algorithms
Linguistic Metrics Synonyms
Syntactic Metrics "is-a" taxonomy
Weigths
Mappings
1st threshold
2nd threshold
accurate domain ontologies and annotated BPMN models

Figure 4.5: Flow chart of matching process for suggesting most accurate domain ontologies and/or accurate annotated BPMN model

The system compares the activated activity label with all concepts of each domain ontology and existing annotated BPMN model in database using sequentially all the string matching algorithms with the fixed weights of each one (bi-gram 0.5, tri-gram 0.5, Levenstein 0.7, Jaro-Winkler 0.8, Jaccard 0.8, char frequency 0.8, soundex 0.5), linguistic and syntactic metrics. The final similarity score is a weighted average of above similarity measures which are given from the following formula (7) and is described with more details in section 4.4.

$$sim = \frac{\sum_{n=0}^{i} sim_{i} * w_{i}}{\sum_{n=0}^{i} w_{i}} \tag{7}$$

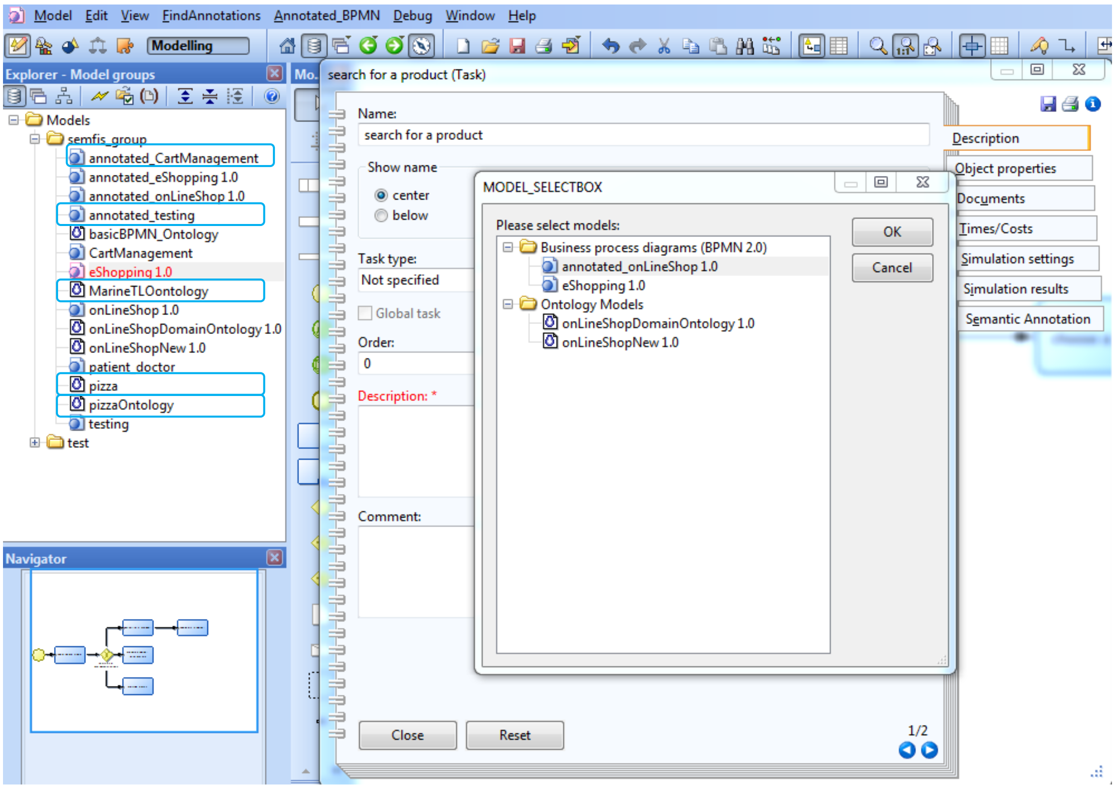
where **sim** = {sim_1,sim_2, ... , sim_i} are the calculated value similarities from all matching algorithms and **W** = {w_1, w_2, ... , w_i} are the corresponding weights of each matching algorithm which are fixed from the system whereas **n** is the number of similarity measures which are used in our implementation.

The same process is followed for all the matching pairs between the loaded activity label and each of the loaded concept of each domain ontology and existing annotated BPMN model in database. The similarity scores are stored in a similarity matrix.

In the next step of matching process, a first fixed threshold (in 0.5) is implemented to the calculated similarity score of each matching pair. If the score is above the given threshold, then the domain ontology concept of this pair is considered an accurate match and might be added into the list with results. This list is sorted from the concepts with the highest score to lowest one (that is from 1.0 to 0.5).

The following step of matching process implements a second fixed threshold (in 0.7). This threshold is also indicative and emerges from in-depth testings at different results every time. In this phase, the system checks the similarity score of the first concept in the above list and if it is beyond the given (0.7) threshold, then the domain ontology or annotated BPMN model, in which this concept is subject to, is considered accurate and it is added in the final list with the most accurate domain ontologies and annotated BPMN models.

In the Figure 4.6, we can see the suggested domain ontologies, as well as the suggested annotated BPMN models for annotating the activities of the "eShopping 1.0" BPMN process model. Specifically, in this example, we annotate the task "search for a product". While drawing the process activity and before the process of its name is completed, the system searches its database to verify if there are accurate domain ontologies for this activity or even already semantically annotated BPMN models with the same concepts. If there are, the system displays them in a model select box and the user can select one of them, either an annotated BPMN model or an ontology. In our example, it has found that there is an accurate annotated BPMN model named "annotated_onLineShop 1.0" and two accurate domain ontologies named "onLineShopDomainOntology 1.0" and "onLineShopNew 1.0".



search for a product (Task)

Name: search for a product

Show name
● center
○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

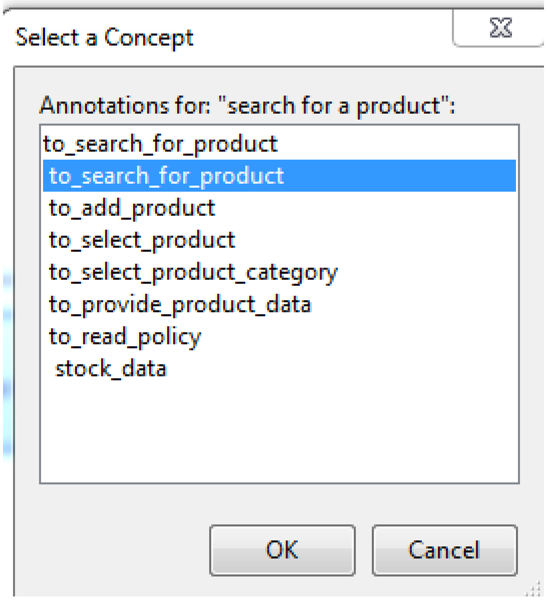
MODEL_SELECTBOX

Please select models:
- Business process diagrams (BPMN 2.0)
  - annotated_onLineShop 1.0
  - eShopping 1.0
- Ontology Models
  - onLineShopDomainOntology 1.0
  - onLineShopNew 1.0

Figure 4.6: The user can select one of the accurate domain ontologies or one of the accurate annotated BPMN elements. Here he has selected the annotated_onLineShop 1.0

As we can notice on the left side of the Figure, inside the light turquoise rounded rectangles, in the "Explorer" of SeMFIS tool, there are also other annotated BPMN models (annotated_CartManagement, annotated_testing), as well as other ontologies (MarineTLOontology, pizza, pizzaOntology) in the SeMFIS database but none of them has related subjects (namely concepts with a score over of 0.7) and for this reason they are not being displayed in the model select box.

If the user selects the annotated BPMN model "annotated_onLineShop 1.0" for the automatically semantic annotation of the current BPMN activity, the system will display the most accurate semantic annotations of this process diagram, as shown in Figure 4.7. The list is ranked from the annotation with the highest score ("to_search_for product") to the annotation with the lowest score ("stock_data").



Select a Concept

Annotations for: "search for a product":

to_search_for_product
to_search_for_product
to_add_product
to_select_product
to_select_product_category
to_provide_product_data
to_read_policy
stock_data

Figure 4.7: In this box, only the most accurate concepts of the selected annotated BPMN model are appeared and the user can select one of them

Comparing the Figures 4.7 and 4.2 will notice that the accurate concepts which are suggested by the selected annotated model are much less from them which are suggested by the selected domain ontology meaning that the user gains in time and effort since the system does not need to search through the entire sum of concepts of a domain ontology but only between the concepts which have already been annotated in the selected annotated BPMN model.

Chapter 5

Presentation of new functionalities of SeMFIS tool

In this chapter, we focus on new functionalities we added as part of our investigation in the modeling environment of SeMFIS tool which are: (1) automatically semantic annotations, (2) RDF/OWL export and (3) querying. In summary, the new functionalities of SeMFIS tool should assist the end user in:

- Adding semi-automatically semantic annotations on existing BPMN process models, that is adding linkable references to domain ontology concepts, and subsequently creating new annotated BPMN models, keeping intact the original ones. These semantic tags assign the important information to perform an early analysis of the process in order to find critical patterns and can be used to guide the user to recognize problems at design time and features which can be useful in further refinements of the process specification [36].
- Exporting the existing ontologies in database and the annotated BPMN models that are created in RDF/OWL format.
- Querying the existing semantically annotated BPMN models, mainly for reusing them or fragments of them which will decrease the effort and time required for modeling of any new processes.

In the following sub-sections we provide a detailed account of each one of the three new functionalities of SeMFIS tool, preceded by a brief description of the SeMFIS architecture tool.

5.1. Brief Description of Architecture of SeMFIS tool

SeMFIS tool [30] has been implemented using the Microsoft Windows-based ADOxx meta modeling platform. ADOxx is professionally developed by BOC Group, a spin-off of the University of Vienna. An overview of the architecture of the current version of SeMFIS tool is shown in Figure 5.1.

At the bottom rests the repository with the modeling subsystem and the Microsoft SQL Server relational database. The modeling subsystem is a Microsoft

Windows application written in C++ and is responsible for handling the persistence of the models in the database, for executing ADOscript statements and for managing the user authentication.

On top of the repository there are the application components for (1) modeling, which handles the model editors, (2) analysis, which provides the AQL query functionality, (3) the web service interface, which accepts SOAP calls containing ADOscript statements and (4) import/export of model information via XML/ADL format. The protégé plugin is integrated via the XML interface of the import/export application component.

Finally, the user interaction is accomplished through a standard Microsoft Windows desktop user interface.

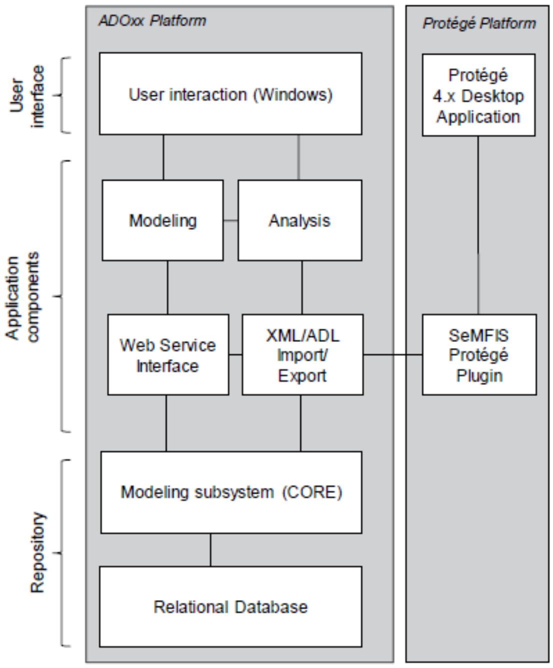


Figure 5.1: SeMFIS Architecture based on ADOxx Platform

We have extended SeMFIS features and specifically Modeling, Analysis and Import/Export functionalities adding characteristics of Semantic Technologies like ontologies, repositories and SPARQL queries. We managed it using ADOscript statements of ADOxx community, the Java language in NetBeans IDE Platform and the Virtuoso - Openlink server. In order to connect with Virtuoso we used Jena jars and Sesame libraries, whereas to convert XML format to OWL one we used jfact-1.2.1.jar, owlapi-distribution-3.5.0.jar, owlexplanation-1.1.2.jar and telemetry-1.0.0.jar libraries. Virtuoso had also to be installed locally to our computer.

5.2. Semi-Automated Suggestions for Ontology-Based Semantic Annotation of BPMN process models

Semantically annotating business processes [11] is focusing on enriching the elements of the process with concepts taken from a domain ontology either already available in SeMFIS database or is created/ updated for the specific process domain by ontology experts. Semantic annotations can be used to augment business process models with different information. In our work, they are used to define the domain semantics of the activities of a BPMN process diagram, meaning that they are used to characterize the nature of each process activity according to a domain ontology. Figure 5.2 depicts an excerpt of a semantically annotated BPMN model[2], where the activities of a business process model have been enriched with automatically semantic annotations.

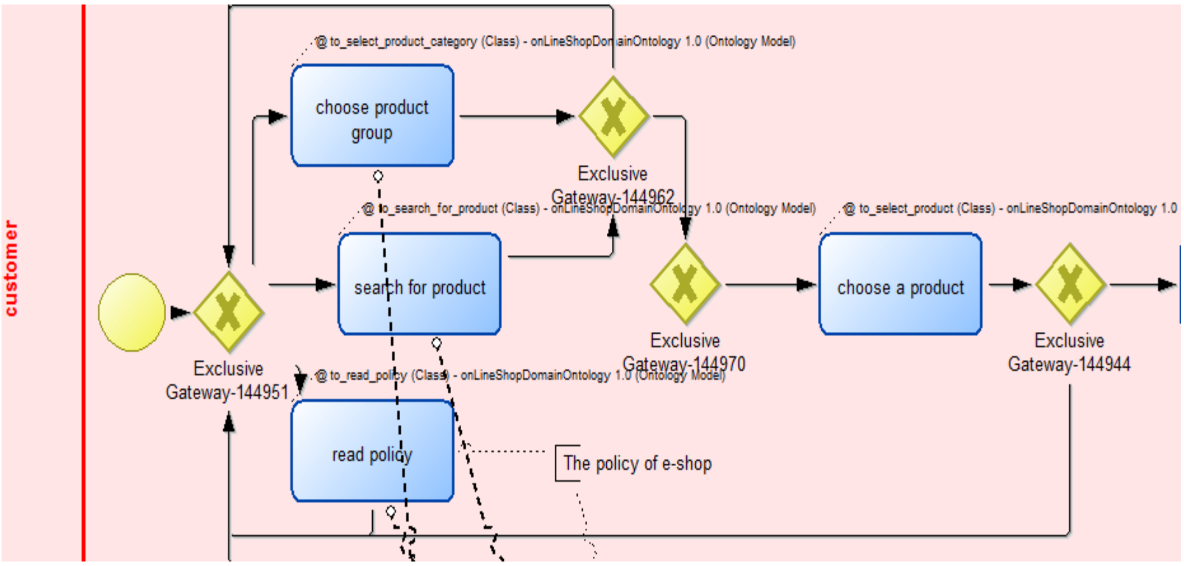

customer
@ to_select_product_category (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
choose product group
Exclusive Gateway-144962
@ to_search_for_product (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
@ to_select_product (Class) - onLineShopDomainOntology 1.0
search for product
Exclusive Gateway-144970
choose a product
Exclusive Gateway-144944
Exclusive Gateway-144951
@ to_read_policy (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
read policy
The policy of e-shop

Figure 5.2: Excerpt of a semantically annotated BPMN model

Semantic annotations are linkable, meaning that if you click on them, they will transfer you in the specific concept of the domain ontology which they were exported from. Any semantic annotation is proceeded by the symbol "@" in order to separate it from plain text annotations which start with the symbol "[". Specifically, the syntax of a semantic annotation is specified by the symbol "@", followed by an annotation term (domain concept) and then by the name and - if existing - by the version of the domain ontology, which was preselected for the annotation process.

<< @ Annotation Term (Class) - Name and Version of Domain Ontology (Ontology Model) >>

[2] The example of "on-Line Shop" business process has been copied from http://selab.fbk.eu/OnLineShop/

Using the modeling component of SeMFIS tool, a business analyst can create a business process diagram based on BPMN 2.0 Specification as shown in Figure 5.3.

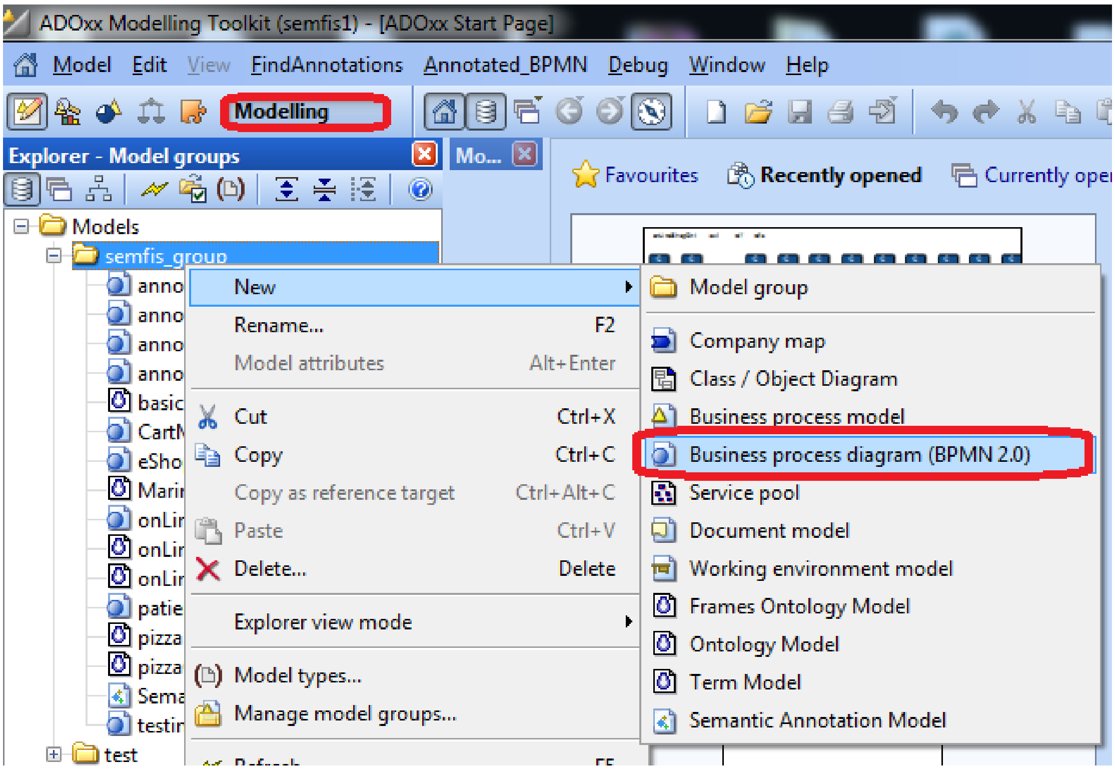
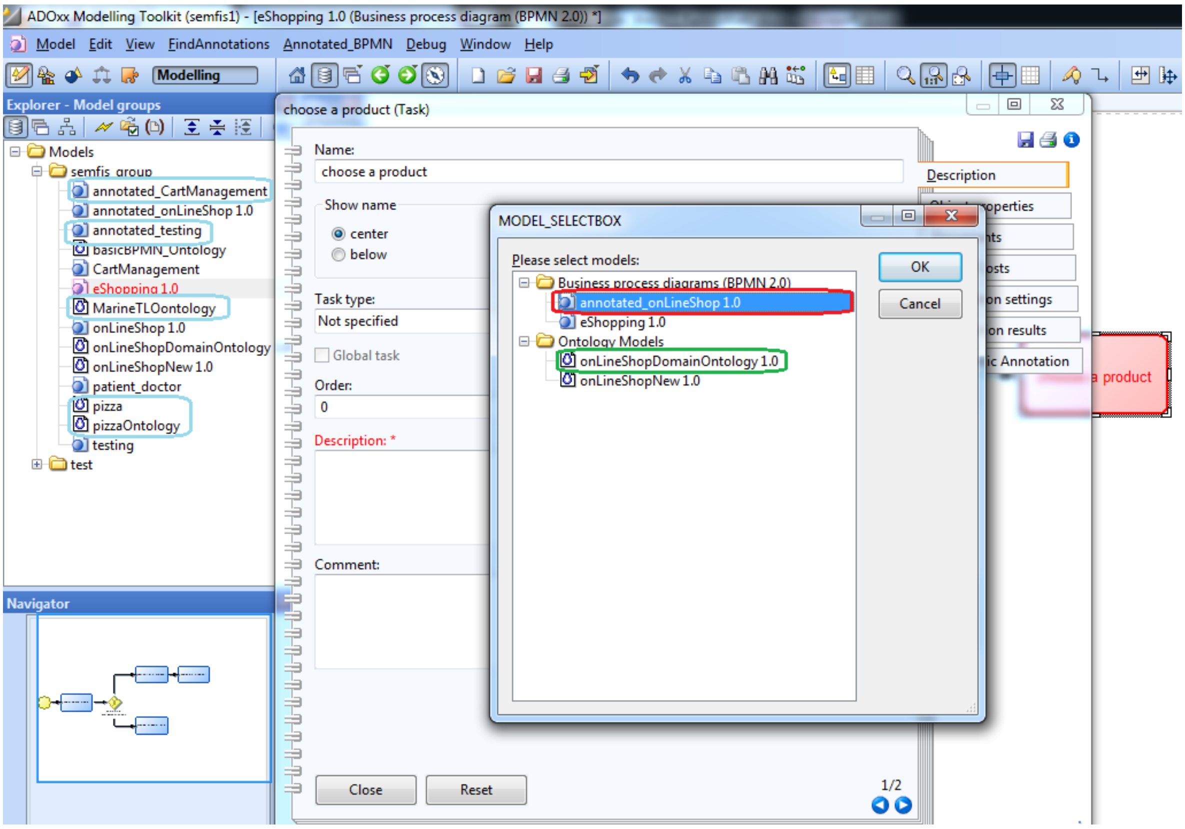


Figure 5.3: Creating a business process diagram based on BPMN 2.0. Specification

While drawing a process activity and before the process of its name is completed, the system searches its database to verify if there are accurate domain ontologies for this activity or even already semantically annotated BPMN models with the same concepts. If there are, the system displays them in a model select box and the user can select one of them, either an annotated BPMN model or an ontology. For example, regarding the activity "choose a product" of BPMN process diagram "e-shopping" in Figure 5.4, the user can select either the already semantically annotated BPMN model "annotated_onLineShop 1.0" or a domain ontology between the "onLineShopDomainOntology 1.0" and the "onLineShopNew 1.0" of the model select box. As we can notice on the left side of the figure, inside the light turquoise rounded rectangles, in the "Explorer" of SeMFIS tool, there are also other annotated BPMN models (annotated_CartManagement, annotated_testing), as well as other ontologies (MarineTLOontology, pizza, pizzaOntology) in the SeMFIS database but none of them has related subjects and for this reason they are not being displayed in the model select box. The procedure on how the system selects the annotated BPMN models and the ontologies where the former will be displayed on the model select box has previously been described in the section 4.5. If there are not accurate annotated BPMN models or ontologies, the system proceeds the process diagram, leaving the user to draw the next desirable BPMN element. The next time where a user will use the system or whenever he/she clicks on the button "FindAnnotations" of the top menu, it automatically searches for all not semantically annotated BPMN models in case a candidate ontology has been created or has been imported in the database or even if an existing ontology in the database has been updated with candidate concepts. Then the system searches so as to find any correlations between these BPMN models and detects which are the most associated diagrams with the concepts of the new ontologies or the updated ones. Afterwards, the system presents them in a model select box where the user will select the diagram which he/she wants to annotate and the semantic annotation process starts automatically. If the system cannot find a suggestion for an activity label, the user can search manually for semantically annotated this activity label, as described below and shown in Figure 5.8. Then the automatically semantic annotation process is continued until all the activity labels are annotated. In the end, the system asks the user if he/she wants to create the corresponding semantically annotated BPMN model. If he/she clicks "ok", the new, semantically annotated BPMN model is created in the SeMFIS database, keeping the original one as well. If the user clicks "cancel", he/she will be able to create the semantically annotated model at any other time he/she wishes clicking on the "Annotated_BPMN" button of the top menu, as described below.



Name: choose a product

Show name
● center
○ below

Task type: Not specified

□ Global task

Order: 0

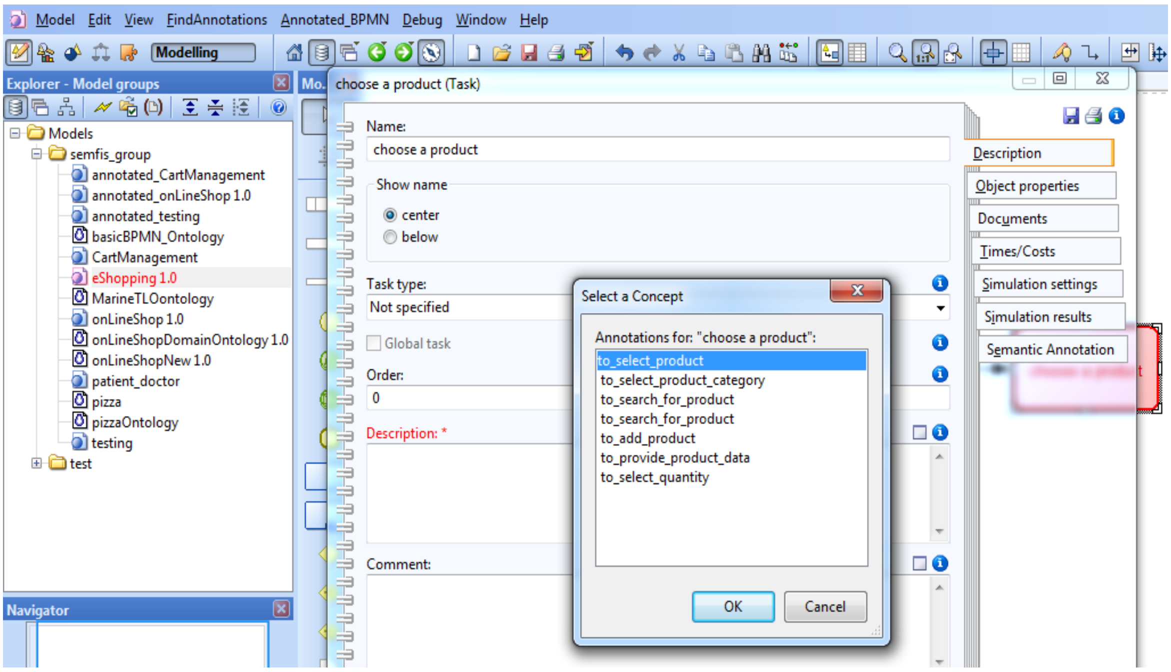
Description: *

Comment:

Please select models:
- Business process diagrams (BPMN 2.0)
  - annotated_onLineShop 1.0
  - eShopping 1.0
- Ontology Models
  - onLineShopDomainOntology 1.0
  - onLineShopNew 1.0

Figure 5.4: Selecting an accurate annotated BPMN model or ontology of SeMFIS database

If the user selects an annotated BPMN model for the automatically semantic annotation of the current BPMN activity, as shown in Figure 5.4 in the red rounded rectangle, the system will display the most accurate semantic annotations of this process diagram, as shown in Figure 5.5.



choose a product (Task)

Name: choose a product

Show name
● center
○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

Select a Concept

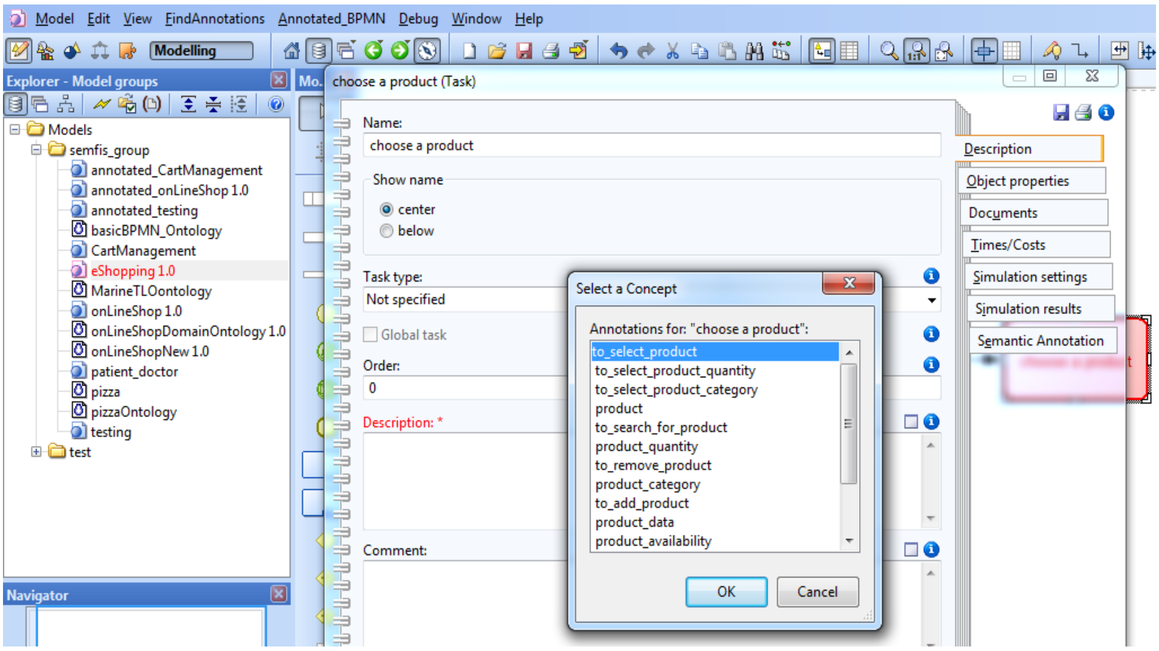
Annotations for: "choose a product":

to_select_product
to_select_product_category
to_search_for_product
to_search_for_product
to_add_product
to_provide_product_data
to_select_quantity

Figure 5.5: Selecting semantic annotation from already annotated BPMN model

The advantage of choosing an already annotated BPMN model is that the user gains in time and effort since the system does not need to search through the entire sum of concepts of a domain ontology but only between the concepts which have already been annotated in the selected annotated BPMN model.

If the user selects an ontology for the automatically semantic annotation of the current BPMN activity, as shown in Figure 5.4 in the green rounded rectangle, the system will display the most accurate semantic annotations (according the similarity measures which have already been described in the section 4.4) of this ontology, as shown in Figure 5.6.



Models
- semfis_group
  - annotated_CartManagement
  - annotated_onLineShop 1.0
  - annotated_testing
  - basicBPMN_Ontology
  - CartManagement
  - eShopping 1.0
  - MarineTLOontology
  - onLineShop 1.0
  - onLineShopDomainOntology 1.0
  - onLineShopNew 1.0
  - patient_doctor
  - pizza
  - pizzaOntology
  - testing
- test

choose a product (Task)

Name: choose a product

Show name: ● center ○ below

Task type: Not specified

□ Global task

Order: 0

Description: *

Comment:

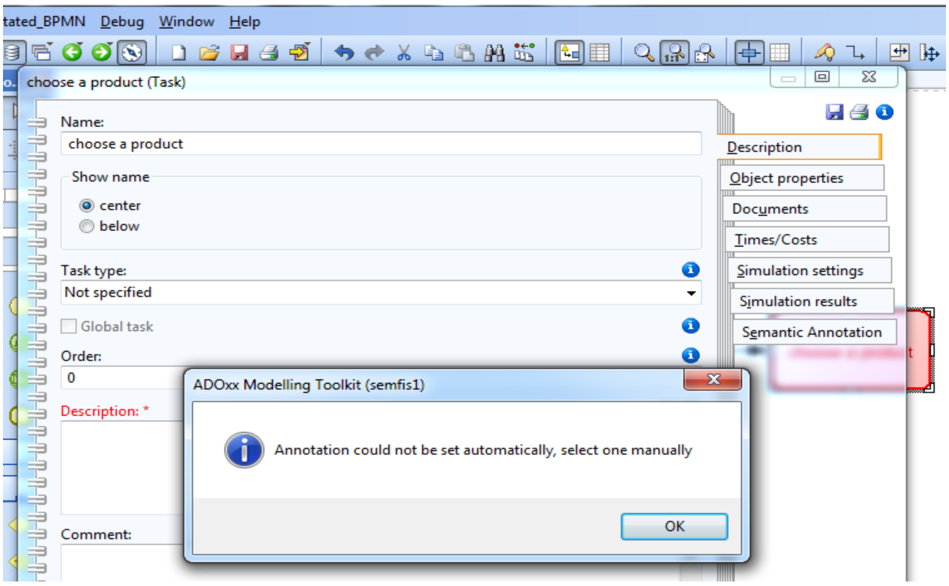
Select a Concept

Annotations for: "choose a product":

to_select_product
to_select_product_quantity
to_select_product_category
product
to_search_for_product
product_quantity
to_remove_product
product_category
to_add_product
product_data
product_availability

Figure 5.6: Selecting annotations from a selected domain ontology

In the case where the user cannot find an accurate semantic annotation of the recommended list, the system gives him/her the opportunity to search manually for semantic annotations from any domain ontology he/she wishes. Specifically, the user will have to click the "Cancel" button and a warning message for manually searching will be displayed on the monitor, as shown in Figure 5.7. Then, by clicking on the "ok" button, the "notebook" of the current BPMN activity opens in the "Semantic Annotation" Chapter, as shown in Figure 5.8. Clicking on the "blue cross" in the "Semantic annotation" box, a window opens where the user can select an ontology from the "Target model" list (left side of the window) and then an object of the selected ontology from the "Target object" list (right side of the window). Finally, if he/she clicks on "add reference" button, the selected concept is added as semantic annotation in the current BPMN activity. Clicking on the "apply" button, the whole procedure is completed and the semantic annotation is displayed above the current BPMN activity in the monitor, as shown in the Figure 5.9.



choose a product (Task)

Name:
choose a product

Show name
● center
○ below

Task type:
Not specified

☐ Global task

Order:
0

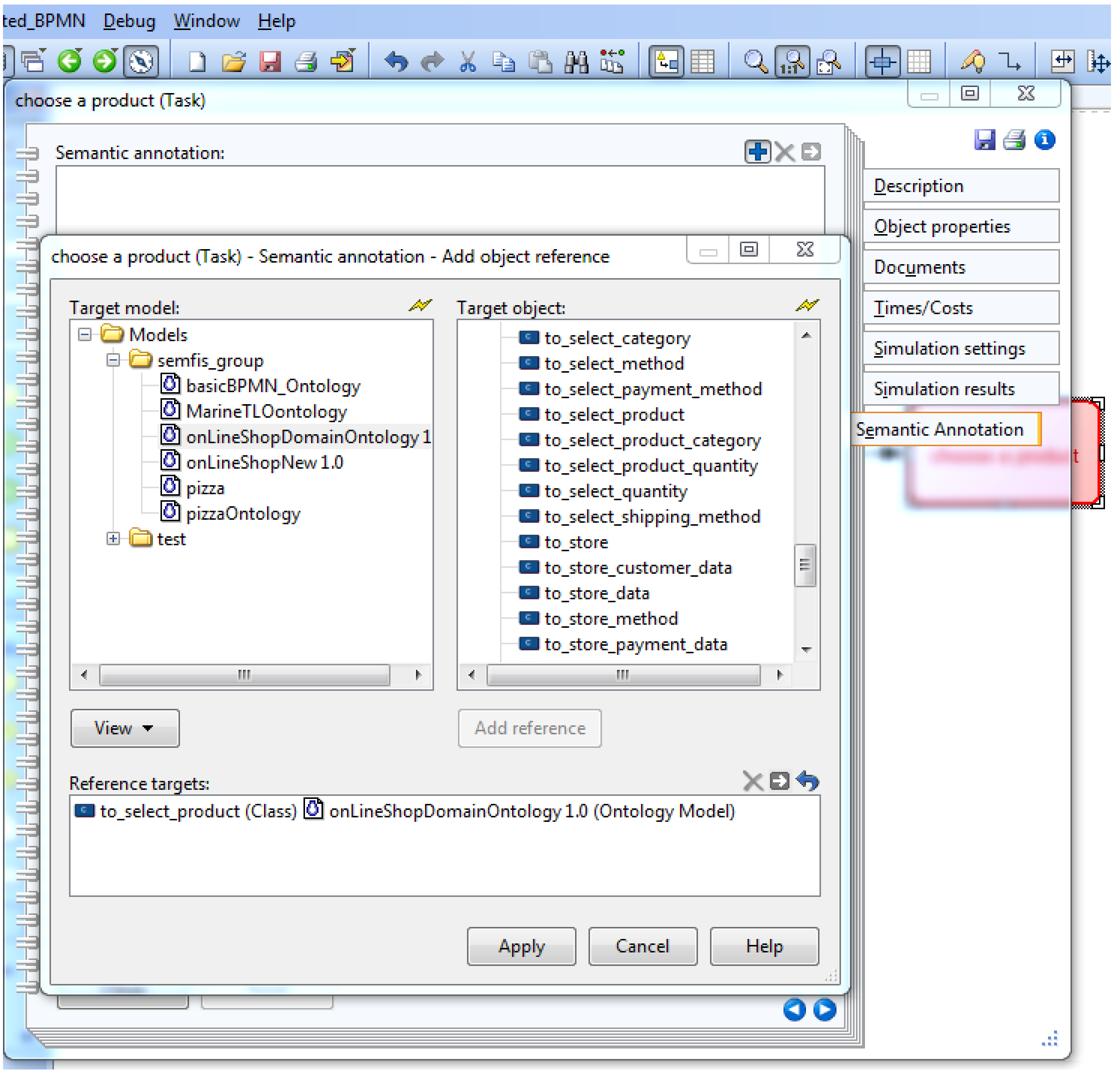
Description: *

Comment:

ADOxx Modelling Toolkit (semfis1)

Annotation could not be set automatically, select one manually

Figure 5.7: Warning message for manually searching semantic annotations



choose a product (Task)

Semantic annotation:

choose a product (Task) - Semantic annotation - Add object reference

Target model:
- Models
  - semfis_group
    - basicBPMN_Ontology
    - MarineTLOontology
    - onLineShopDomainOntology 1
    - onLineShopNew 1.0
    - pizza
    - pizzaOntology
  - test

Target object:
- to_select_category
- to_select_method
- to_select_payment_method
- to_select_product
- to_select_product_category
- to_select_product_quantity
- to_select_quantity
- to_select_shipping_method
- to_store
- to_store_customer_data
- to_store_data
- to_store_method
- to_store_payment_data

Reference targets:
to_select_product (Class) onLineShopDomainOntology 1.0 (Ontology Model)

Figure 5.8: Manually selection of semantic annotations

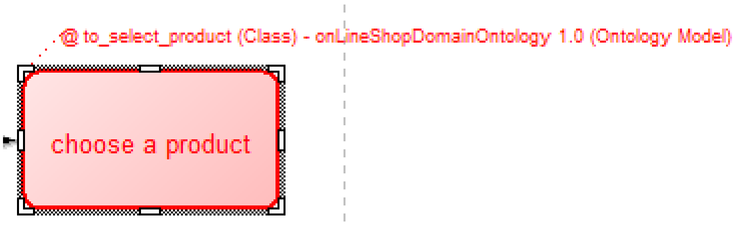

@ to_select_product (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
choose a product

Figure 5.9: Semantic annotation in the BPMN task "choose a product"

As it already has been indicated, if the user clicks on the semantic annotation, in our example on the annotation "@to_select_product (Class) - onLineShopDomainOntology 1.0 (Ontology Model)", the system will transfer him/her in the specific concept of the selected ontology, as shown in the Figure 5.10.

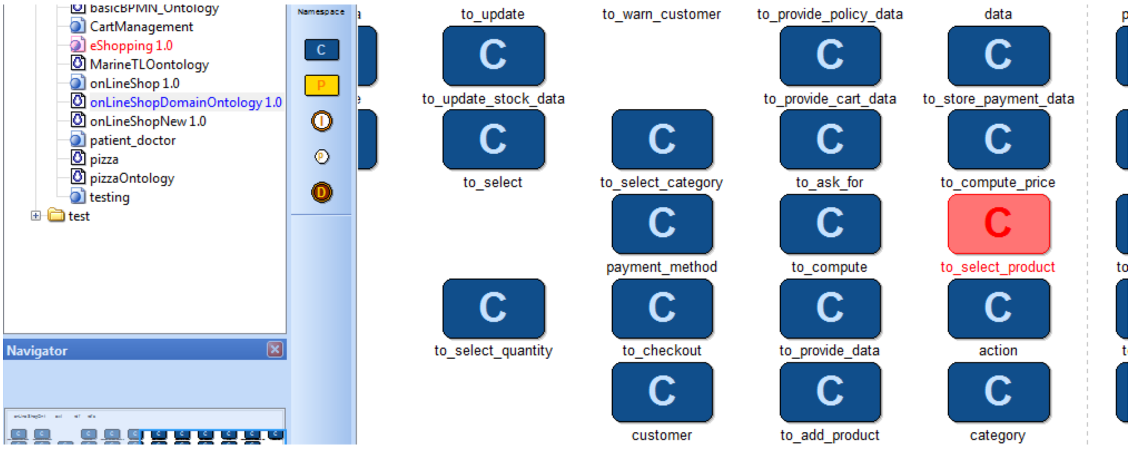


Figure 5.10: Transfer on the specific concept of the selected ontology after clicking on the semantic annotation "to_select_product"

When the user finishes the process diagram or whenever he/she wants during drawing the diagram, he/she can click the "Annotated_BPMN" button from the top menu and the system automatically produces the corresponding annotated BPMN model. With this way, business analysts have both the original process diagram (without semantic annotations) and the corresponding annotated diagram with all the selected semantic annotations, as shown in the Figure 5.11. If the user decides to change the label of an already existing activity, then the system searches for new semantic annotation for this activity and the annotated model is updated automatically with the new semantic annotation either the change is made to the original model or to the corresponding annotated one. The new label of the already existing activity is updated in both models at the same time. If the user has left the original diagram incomplete and decides to complete it another time, the system again searches for semantic annotations for the new activity labels which are added in the original model. In order to transfer them in the corresponding annotated model, the user has to click on the "Annotated_BPMN" button and the corresponding annotated model is updated with the new BPMN elements, as well as the new semantic annotations which disappear from the original one.

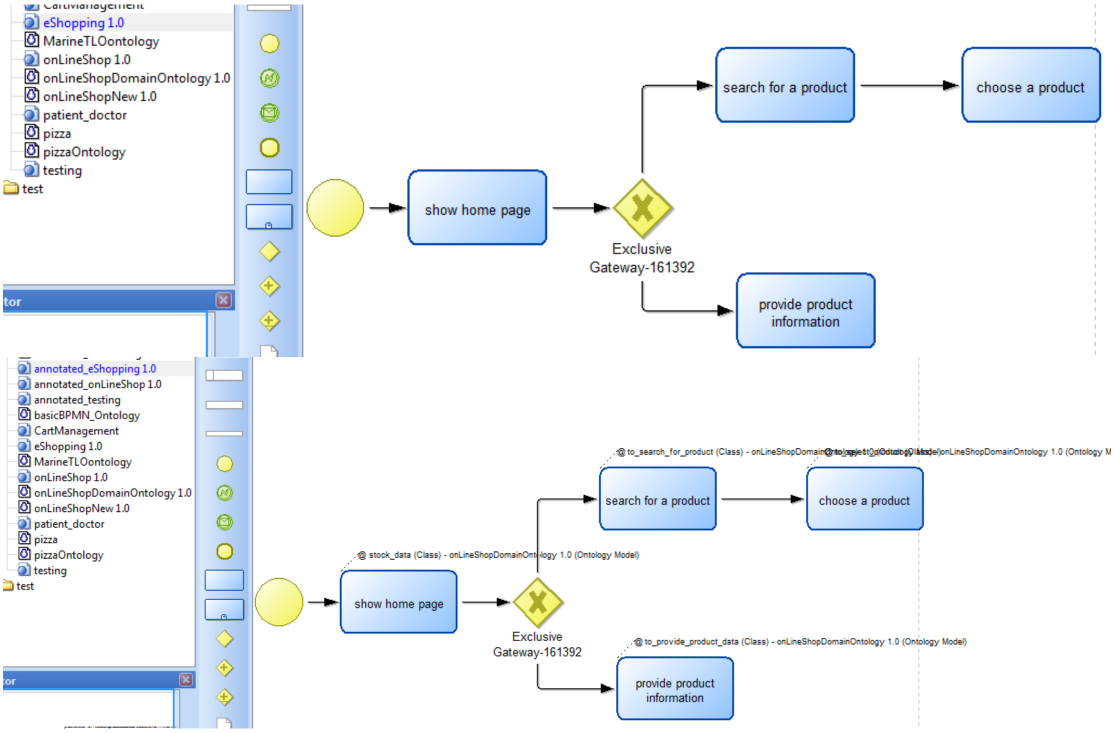

CartManagement
eShopping 1.0
MarineTLOontology
onLineShop 1.0
onLineShopDomainOntology 1.0
onLineShopNew 1.0
patient_doctor
pizza
pizzaOntology
testing
test
show home page
Exclusive Gateway-161392
search for a product
choose a product
provide product information
annotated_eShopping 1.0
annotated_onLineShop 1.0
annotated_testing
basicBPMN_Ontology
to_search_for_product (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
stock_data (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
to_provide_product_data (Class) - onLineShopDomainOntology 1.0 (Ontology Model)

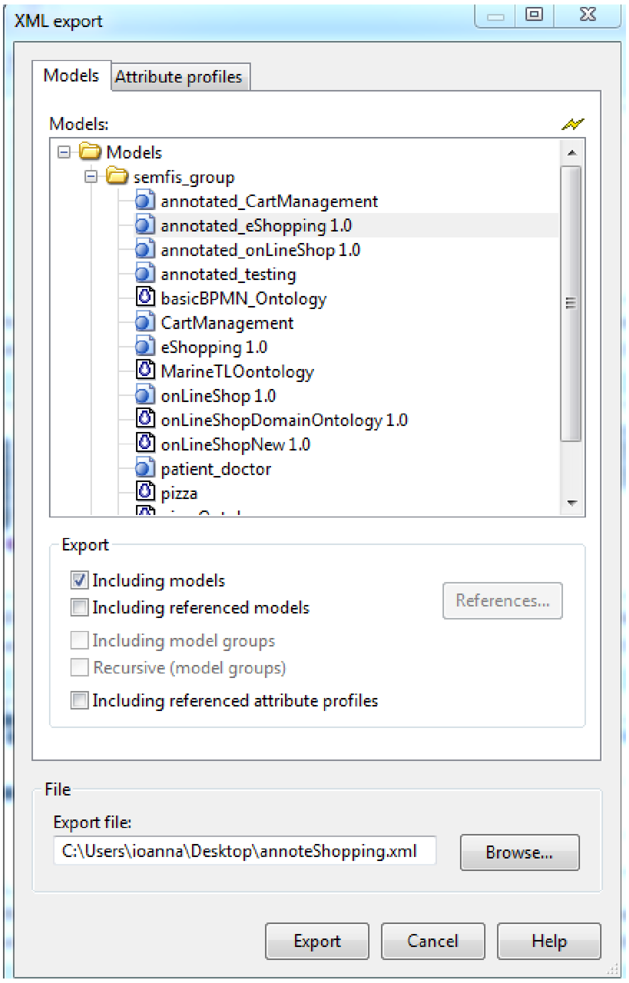
Figure 5.11: The original BPMN diagram "eShopping 1.0" (top) and the corresponding annotated diagram "annotated_eShopping 1.0" (bottom)

5.3. OWL / RDF Export

SeMFIS tool [30] provides import and export interfaces, via the application component for import/export, for exchanging model information in ADL and XML formats. In addition, it has integrated a Protégé plugin via the XML interface for importing owl ontologies from the Protégé platform. The XML import/export is used to exchange information from arbitrary model types. It is therefore more generic and well suited for exchanging information with other tools and platforms. On the other hand, the ADL import/export interface is used for exchanging information with other ADOxx based tools that do not offer an XML interface.

At this moment, the SeMFIS tool does not support an OWL/RDF export interface for exchanging information with tools and platforms which support owl/rdf formats and do not offer an XML interface. Therefore, integrating OWL/RDF export interface in the SeMFIS tool, will help it to collaborate with platforms and repositories which support owl format, allowing better understanding, documenting and querying on semantically annotated BPMN models, as will see in the next section.

The OWL/RDF export functionality has been added to import/export component of SeMFIS tool, in "model" menu (top, left menu) under the XML export. In the first step, when the user clicks on it, an XML export window appears in the monitor as shown in the Figure 5.12. The user can select to export either an ontology or a semantically annotated BPMN model. If the last one includes sub-models, the user has to check the command "including referenced models" in order to be integrated in the xml file. In the frame "Export file", the user selects where to save the xml file and subsequently its name. Finally, clicking on "Export" button, an xml file is created in the path which has been defined by the user and an information box appears for the exported file.


XML export

Models:

- Models
  - semfis_group
    - annotated_CartManagement
    - annotated_eShopping 1.0
    - annotated_onLineShop 1.0
    - annotated_testing
    - basicBPMN_Ontology
    - CartManagement
    - eShopping 1.0
    - MarineTLOontology
    - onLineShop 1.0
    - onLineShopDomainOntology 1.0
    - onLineShopNew 1.0
    - patient_doctor
    - pizza

Export

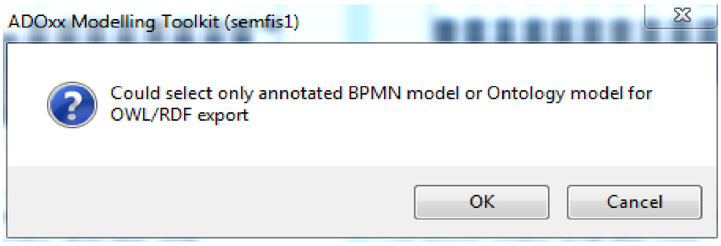
☑ Including models
□ Including referenced models
□ Including model groups
□ Recursive (model groups)
□ Including referenced attribute profiles

File

Export file: C:\Users\ioanna\Desktop\annoteShopping.xml

Figure 5.12: The first step of OWL/RDF export is the xml export of selected ontology or annotated BPMN model

The user cannot export any other model type. In this case, a warning message appears which informs the user that only annotated BPMN models and ontologies can be exported, as shown in the Figure 5.13, and the "XML export" window appears again.



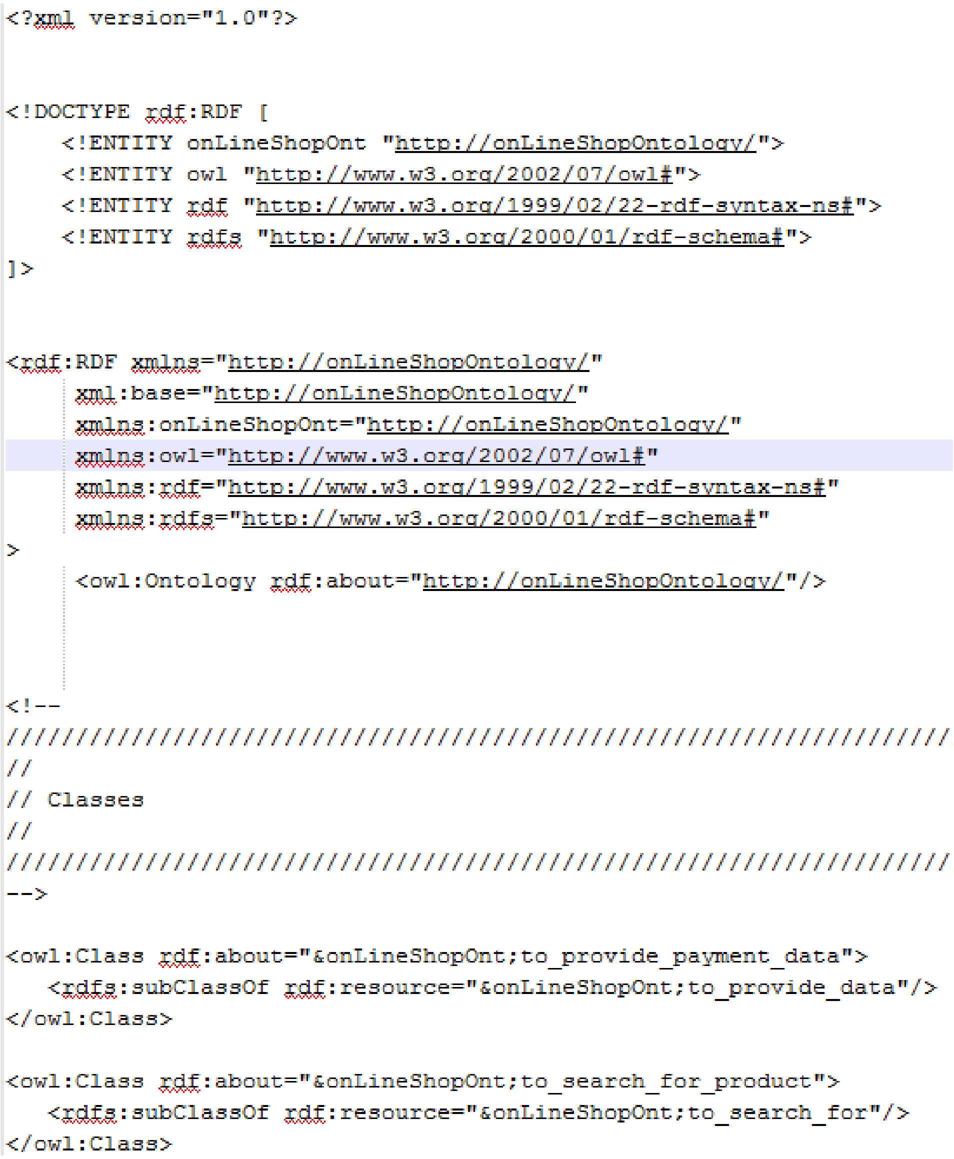
ADOxx Modelling Toolkit (semfis1)

Could select only annotated BPMN model or Ontology model for OWL/RDF export

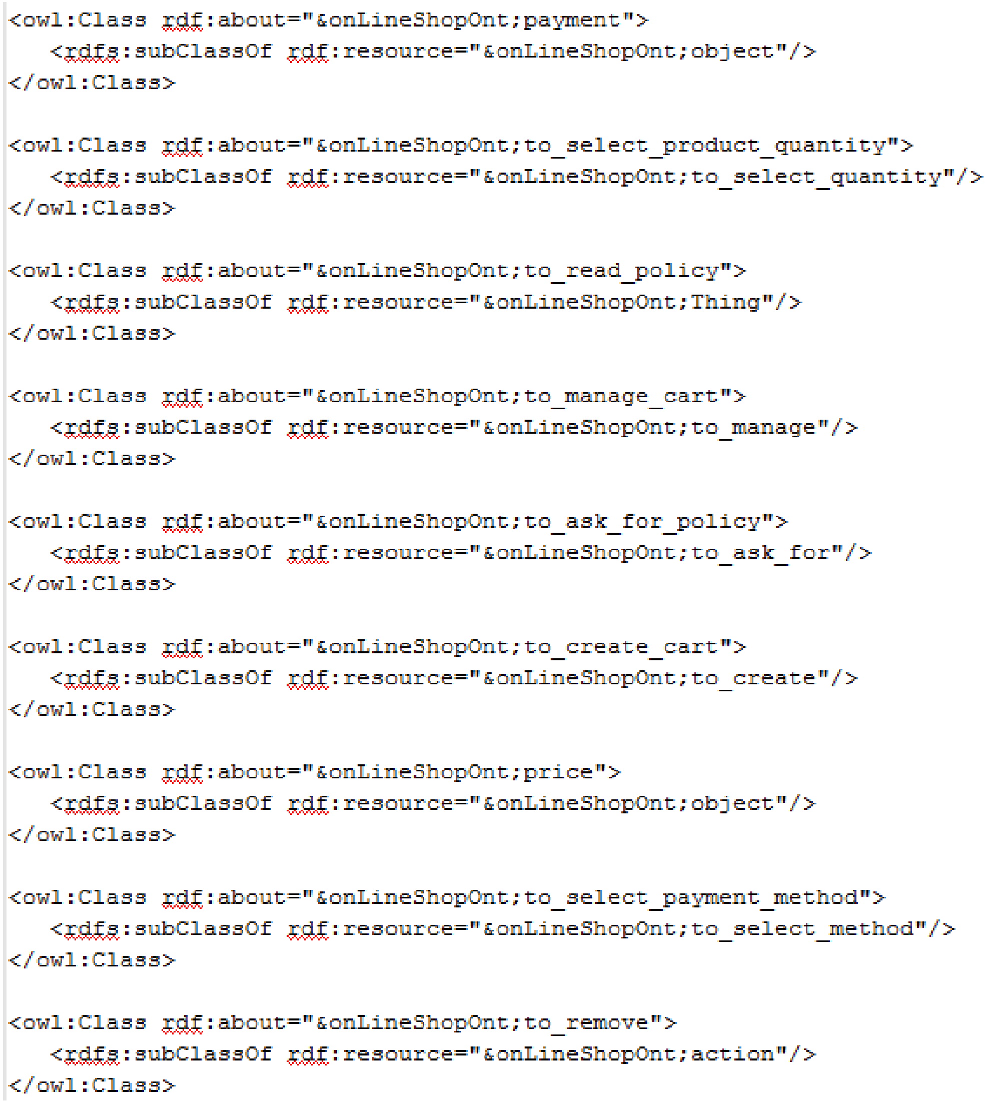
Figure 5.13: Warning message for OWL/RDF export

In the second step, if the user has selected to export an ontology, a message informs him/her about saving ontology in owl format and afterwards the user have to select where to save the owl file adding in the end of its name the suffix ".owl". An excerpt of exporting the "onLineShopDomainOntology" ontology in owl format is shown in Figure 5.14.

If the user has selected to export a semantically annotated BPMN model, except from the xml file which is exported by the user, the system also exports automatically the general ontology "basicBPMN_Ontology" which has been described in chapter 3. Hereafter, the owl file is exported and is saved as described in the above paragraph. The new owl file is the general ontology "basicBPMN_Ontology" enhanced with a set of instances which has been derived from the selected semantically annotated BPMN model. An excerpt of exporting the "Annotated_eShopping 1.0" annotated BPMN model in owl format is shown in Figure 5.15.

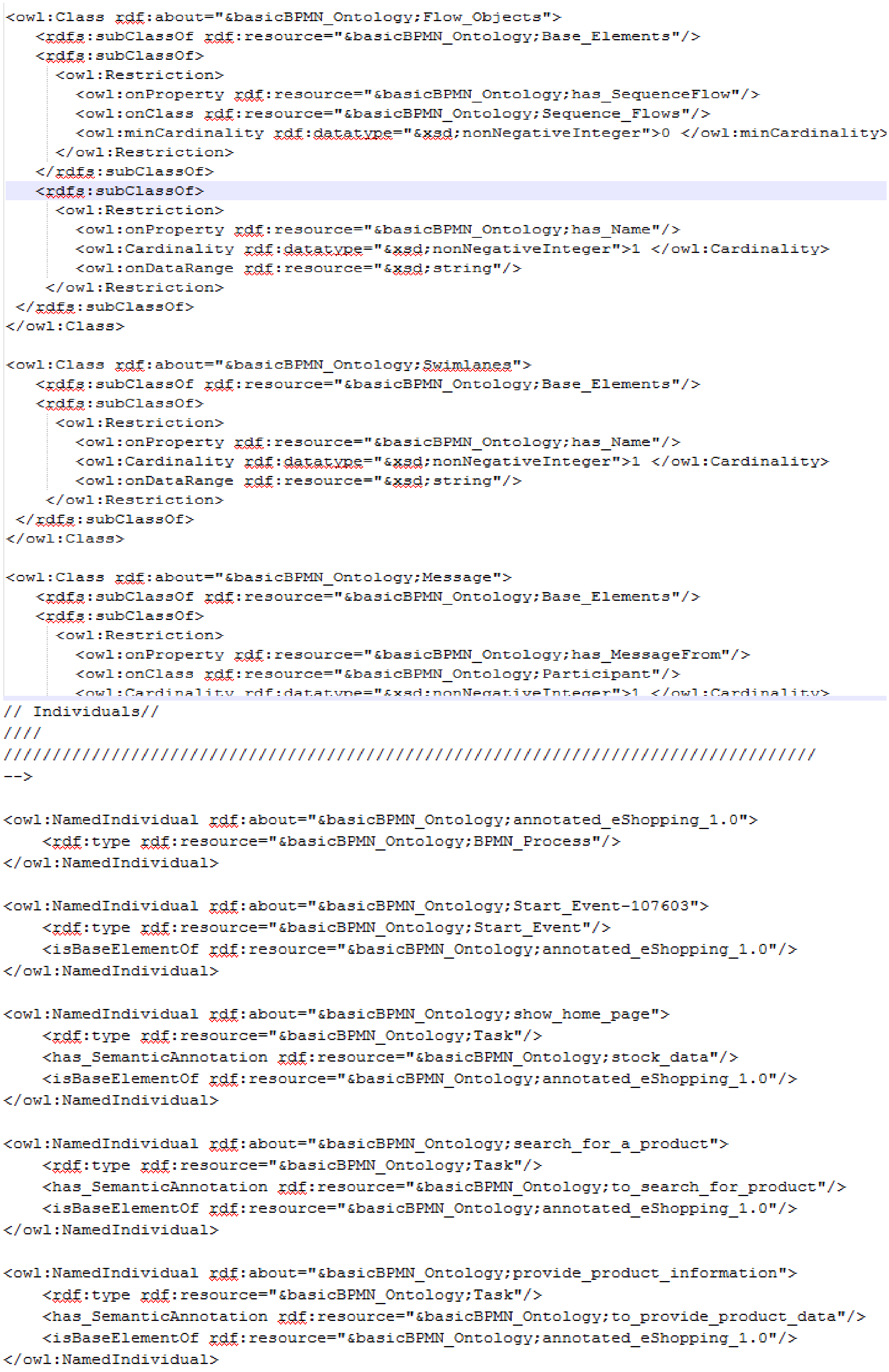


```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY onLineShopOnt "http://onLineShopOntology/">
    <!ENTITY owl "http://www.w3.org/2002/07/owl#">
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
]>


<rdf:RDF xmlns="http://onLineShopOntology/"
     xml:base="http://onLineShopOntology/"
     xmlns:onLineShopOnt="http://onLineShopOntology/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
     <owl:Ontology rdf:about="http://onLineShopOntology/"/>



<!--
////////////////////////////////////////////////////////////////////
//
// Classes
//
////////////////////////////////////////////////////////////////////
-->

<owl:Class rdf:about="&onLineShopOnt;to_provide_payment_data">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_provide_data"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_search_for_product">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_search_for"/>
</owl:Class>
```



```
<owl:Class rdf:about="&onLineShopOnt;payment">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;object"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_select_product_quantity">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_select_quantity"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_read_policy">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;Thing"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_manage_cart">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_manage"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_ask_for_policy">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_ask_for"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_create_cart">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_create"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;price">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;object"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_select_payment_method">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;to_select_method"/>
</owl:Class>

<owl:Class rdf:about="&onLineShopOnt;to_remove">
   <rdfs:subClassOf rdf:resource="&onLineShopOnt;action"/>
</owl:Class>
```

Figure 5.14: Excerpt of "onLineShopDomainOntology" ontology in owl format



```
<owl:Class rdf:about="&basicBPMN_Ontology;Flow_Objects">
   <rdfs:subClassOf rdf:resource="&basicBPMN_Ontology;Base_Elements"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&basicBPMN_Ontology;has_SequenceFlow"/>
       <owl:onClass rdf:resource="&basicBPMN_Ontology;Sequence_Flows"/>
       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">0 </owl:minCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&basicBPMN_Ontology;has_Name"/>
       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger">1 </owl:Cardinality>
       <owl:onDataRange rdf:resource="&xsd;string"/>
    </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&basicBPMN_Ontology;Swimlanes">
   <rdfs:subClassOf rdf:resource="&basicBPMN_Ontology;Base_Elements"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&basicBPMN_Ontology;has_Name"/>
       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger">1 </owl:Cardinality>
       <owl:onDataRange rdf:resource="&xsd;string"/>
    </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&basicBPMN_Ontology;Message">
   <rdfs:subClassOf rdf:resource="&basicBPMN_Ontology;Base_Elements"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="&basicBPMN_Ontology;has_MessageFrom"/>
       <owl:onClass rdf:resource="&basicBPMN_Ontology;Participant"/>
       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger">1 </owl:Cardinality>
```

```
// Individuals//
////
///////////////////////////////////////////////////////////////////////////////////
-->

<owl:NamedIndividual rdf:about="&basicBPMN_Ontology;annotated_eShopping_1.0">
    <rdf:type rdf:resource="&basicBPMN_Ontology;BPMN_Process"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&basicBPMN_Ontology;Start_Event-107603">
    <rdf:type rdf:resource="&basicBPMN_Ontology;Start_Event"/>
    <isBaseElementOf rdf:resource="&basicBPMN_Ontology;annotated_eShopping_1.0"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&basicBPMN_Ontology;show_home_page">
    <rdf:type rdf:resource="&basicBPMN_Ontology;Task"/>
    <has_SemanticAnnotation rdf:resource="&basicBPMN_Ontology;stock_data"/>
    <isBaseElementOf rdf:resource="&basicBPMN_Ontology;annotated_eShopping_1.0"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&basicBPMN_Ontology;search_for_a_product">
    <rdf:type rdf:resource="&basicBPMN_Ontology;Task"/>
    <has_SemanticAnnotation rdf:resource="&basicBPMN_Ontology;to_search_for_product"/>
    <isBaseElementOf rdf:resource="&basicBPMN_Ontology;annotated_eShopping_1.0"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&basicBPMN_Ontology;provide_product_information">
    <rdf:type rdf:resource="&basicBPMN_Ontology;Task"/>
    <has_SemanticAnnotation rdf:resource="&basicBPMN_Ontology;to_provide_product_data"/>
    <isBaseElementOf rdf:resource="&basicBPMN_Ontology;annotated_eShopping_1.0"/>
</owl:NamedIndividual>
```

Figure 5.15: Excerpt of "Annotated_eShopping 1.0" annotated BPMN model in owl format

5.4. Sparql Queries

SeMFIS tool has also integrated the analysis component of the ADOxx platform which is used for composing and executing queries expressed in AQL (ADOxx Query Language). AQL is not as powerful language as SQL for relational databases and SPARQL for semantic web. It can be used for easily gathering information from the models in a relational database but not for retrieving data from repositories in the semantic web. Technically speaking, it is a difficult language in learning, so it is limited to a small BPMN community which focuses on applications based on ADOxx platform and has knowledge about ADOscript and therefore about AQL language.

On the other hand, SPARQL is a declarative semantic query language for databases for retrieving and manipulating data stored in RDF format. It was made a standard by the World Wide Web Consortium (W3C) and one of the key technologies of the semantic web. For this reason we decided to use SPARQL query language for retrieving information out of the RDF/OWL files which are exported by the SeMFIS tool, as analyzed in the previous section.

At this point, it is essential to briefly introduce the SPARQL query language. Still, before that, it will come in handy to remember what an RDF Triple is. Assuming there is a pair wise disjoint infinite sets I, B, and L (IRIs, Blank nodes, and Literals, respectively). An RDF triple is a tuple (v1, v2, v3) $\in$ (I $\cup$ B) × I × (I $\cup$ B $\cup$ L) where v1 is the subject and can be IRIs or Blank node, v2 is the predicate and can be IRIs and v3 is the object and can be IRIs, Blank node or literal. Most forms of SPARQL query [58] contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable prefixed by the symbol "?" or "$". A basic graph pattern matches a sub-graph of the RDF data when RDF terms from that sub-graph may be substituted for the variables and the result is RDF graph equivalent to the sub-graph. An RDF graph is a set of RDF assertions, manipulated as a labeled directed graph. So, queries describe sub-graphs of the queried RDF graph. The basic syntax of a SPARQL query is:

Table 5.1: basic syntax of SPARQL queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT clause
(FROM <http://....                    )
WHERE { clause
}
MODIFIER clause
```

The PREFIX keyword associates a prefix label (like rdf:) with an IRI (like <http://www.w3.org/1999/02/22-rdf-syntax-ns#>). Most of the times the above prefixes are the standard predefined namespace prefixes into SPARQL Endpoints for SPARQL queries, so the user does not need to rewrite them. But the user, of course, can add as many prefixes as the vocabularies (ontologies) he/she uses. SELECT clause retrieves variables and their bindings as a table. FROM clause is optional and indicates the sources for the data against which to find matches. WHERE clause defines patterns to match against the data. MODIFIER clause modifies the result set, for example the modifiers ORDER BY, OFFSET or LIMIT re-organize rows whereas GROUP BY combines them.

Let us see an example of a SPARQL query. Suppose we want to find all the Tasks of a process which have semantic annotation with the class "to_search_for_product" as well as the process which it belongs in. The syntax of SPARQL query is:

Table 5.2: Example of a SPARQL query

```
PREFIX basicBPMN_Ontology:
<http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#>

SELECT ?activity ?process
FROM <http://localhost:8890/BPMNontology>
WHERE {
?activity a basicBPMN_Ontology:Task.
?activity basicBPMN_Ontology:has_SemanticAnnotation.
basicBPMN_Ontology:to_search_for_product.
?activity basicBPMN_Ontology:isBaseElementOf ?process.
}
```

As illustrated by the above Table, the prefix "basicBPMN_Ontology" is added by the user to indicate the ontology that contains the patterns of WHERE clause; SELECT clause indicates the variables where the data will be retrieved as a table; the FROM clause gives us a specific graph in which the results will be searched. The WHERE clause instead denotes the triple patterns that match the data we want to retrieve from the specific graph "BPMNontology".

In order to be able to work with SPARQL queries, we had to upload owl files which are exported from the SeMFIS tool in a server which manages RDF data. We selected for this reason the Virtuoso - Open Source because it is a scalable cross-platform server that combines Relational, Graph, and Document Data Management with Web Application Server and Web Services Platform functionality. So, it is a hybrid universal server where triple store access is available via SPARQL.

In analysis component of SeMFIS tool, under the AQL queries, we added the SPARQL queries functionality, as shown in Figure 5.16. Clicking on it, all the annotated BPMN models in the SeMFIS explorer are converted to an RDF/OWL file automatically and then the latter is uploaded to Virtuoso - Open Source server. Subsequently, a box with standardized SPARQL queries appears in the screen giving the user the opportunity to select one of them, as shown in the Figure 5.17. We have made 7 standardized queries with the most useful components for a member of a BPMN community. The 7 standardized queries are:

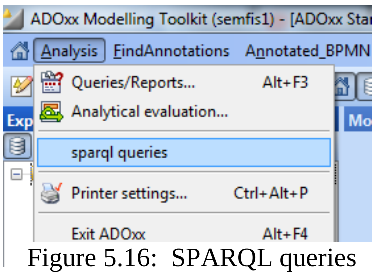
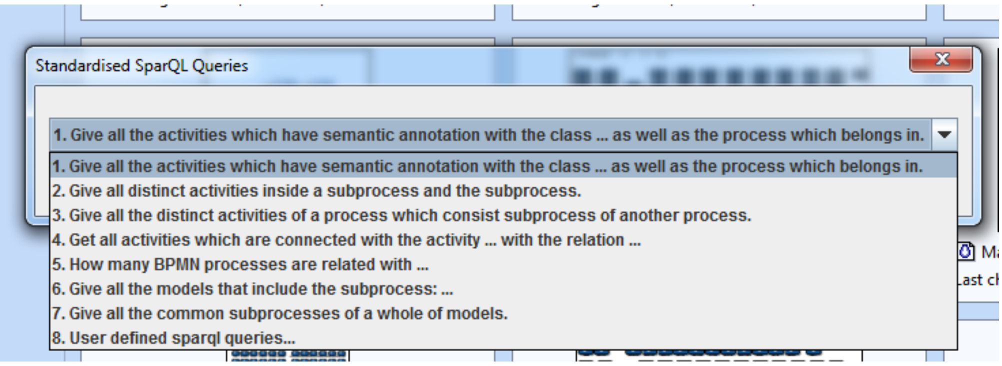


Figure 5.16: SPARQL queries functionality

1. *Give all the activities which have semantic annotation with the class ... as well as the process which belongs in.*
2. *Give all distinct activities inside a sub-process and the sub-process.*
3. *Give all the distinct activities of a process which consist sub-process of another process.*
4. *Get all activities which are connected with the activity ... with the relation ...*
5. *How many BPMN processes are related with ...*
6. *Give all the models that include the sub-process: ...*
7. *Give all the common sub-processes of a whole of models.*

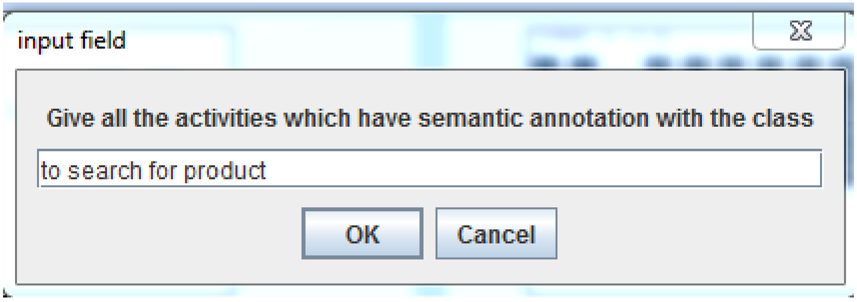


1. Give all the activities which have semantic annotation with the class ... as well as the process which belongs in.

1. Give all the activities which have semantic annotation with the class ... as well as the process which belongs in.
2. Give all distinct activities inside a subprocess and the subprocess.
3. Give all the distinct activities of a process which consist subprocess of another process.
4. Get all activities which are connected with the activity ... with the relation ...
5. How many BPMN processes are related with ...
6. Give all the models that include the subprocess: ...
7. Give all the common subprocesses of a whole of models.
8. User defined sparql queries...

Figure 5.17: Select one of the standardized SPARQL queries or define your own query with the 8 selection

In some of queries, the user has to add additional information on what interests him/her. For example, if the user selects the query 1, an input field asks from the user to add the class which he/she is interested for (e.g. "to search for product"), as shown in the Figure 5.18.

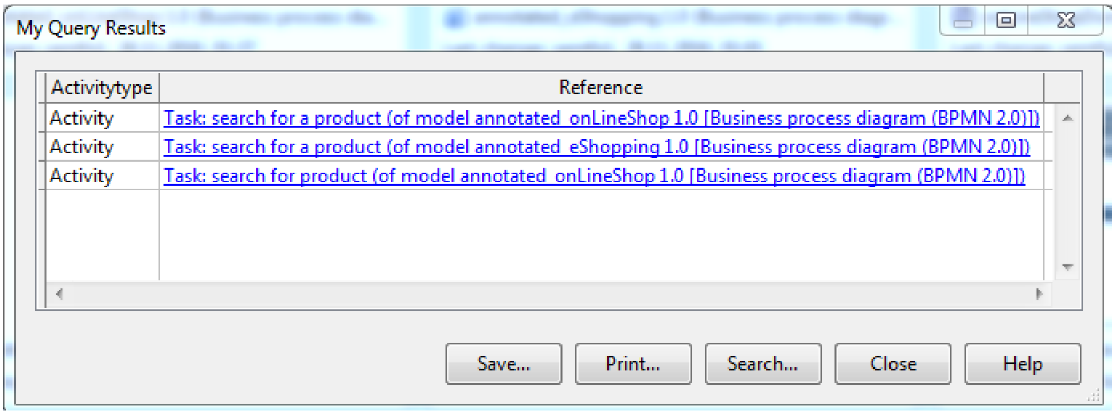


Give all the activities which have semantic annotation with the class

to search for product

Figure 5.18: The user is interested to find all the activities which have semantic annotation with the class "to_search_for_product"

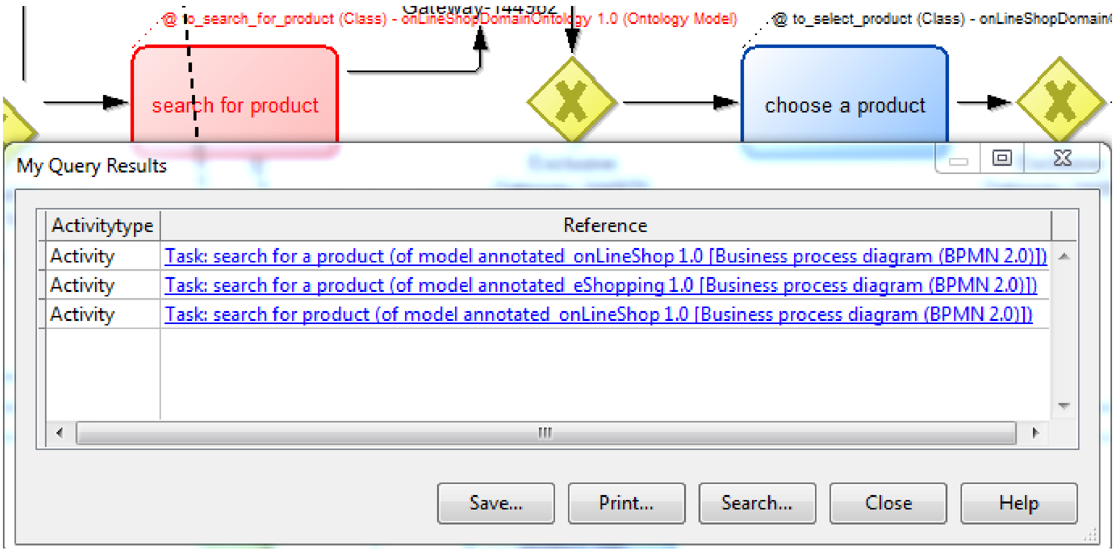
The final results of a query are represented as a table with links where we can see the type of activity (e.g. Task), the activity label (e.g. search for a product) and the model which contains it (e.g. annotated_onLineShop 1.0), as shown in the Figure 5.19.



Activitytype	Reference
Activity	Task: search for a product (of model annotated_onLineShop 1.0 [Business process diagram (BPMN 2.0)])
Activity	Task: search for a product (of model annotated_eShopping 1.0 [Business process diagram (BPMN 2.0)])
Activity	Task: search for product (of model annotated_onLineShop 1.0 [Business process diagram (BPMN 2.0)])

Figure 5.19: Results of the query 1, giving as input field the class "to search for product"

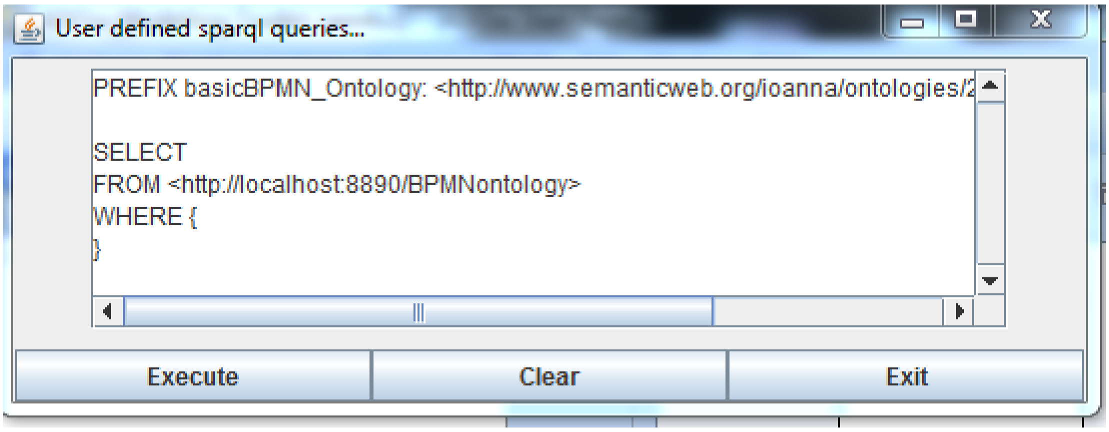
If you click on one of the aforementioned links, it transfers you in the specific activity or process which automatically turns red, as shown in Figure 5.20, where we have clicked on the first result and the system has transfered us on the model "annotated_onLineShop 1.0" and specifically in the red task with the label "search for product".


@ to_search_for_product (Class) - onLineShopDomainOntology 1.0 (Ontology Model)
@ to_select_product (Class) - onLineShopDomain
search for product
choose a product
My Query Results
Save...
Print...
Search...
Close
Help

Activitytype	Reference
Activity	Task: search for a product (of model annotated_onLineShop 1.0 [Business process diagram (BPMN 2.0)])
Activity	Task: search for a product (of model annotated_eShopping 1.0 [Business process diagram (BPMN 2.0)])
Activity	Task: search for product (of model annotated_onLineShop 1.0 [Business process diagram (BPMN 2.0)])

Figure 5.20: Clicking on the first result, it transfers us on the task "search for product" of model "annotated_onLineShop 1.0"

Finally, we give the opportunity to a business member, who has the knowledge, to make his/her own SPARQL query selecting the eighth query called "User defined sparql queries" of the "Standardized SparQL Queries" box. A new frame appears having a predefined prefix for our basic BPMN ontology, as well as the basic syntax of a SPARQL query, as shown in Figure 5.21. We have also predefined the graph where the query will be done in order to save time but the user can delete it and search in each and every one of the graphs of Virtuoso server.



```
PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2

SELECT
FROM <http://localhost:8890/BPMNontology>
WHERE {
}
```

Figure 5.21: The frame where a user can make his/her own SPARQL query

Querying the process space with a machine readable representation of a logical expression which defines a subset of facts of the process space is a very important task. It can help the BPMN community to reuse parts of business process models when creating a new one, to detect cross-process relations and to reinforce the interconnectivity and interoperability saving costs and time when establishing inter-organizational business collaborations.

Chapter 6

Conclusion and Future Work

In this thesis, we presented an approach for the integration of semantic annotations in modeling tools to support the graphical representation of business process models with concepts derived from domain ontologies. These annotations clarify the activity labels of a model by associating it to candidate domain concepts. The goal of semantic annotations is to make these business process models more precise, to increase the refinement of business process modeling and finally to enable querying process model repositories. Moreover, a semantically annotated model is both comprehensible to humans and accessible to machines because of the integration with ontologies.

For this purpose, we have extended the SeMFIS tool, using the freely available ADOxx platform, supporting the implementation of semantic annotations on business process activities (tasks, sub-processes), which have created with BPMN 2.0 language, via semi-automated suggestions. Specifically, at the time where a user designs a business process model, the tool will retrieve and recommend semantic annotations for the activities of this business process model using an ontology based data matching strategy. The user will have the choice either to use one of the automated suggestions or to search manually for it. The matching strategy is based on calculating a combined similarity measure between a BPMN activity label and concepts of a selected domain ontology. The combined similarity measure consists of string algorithms where the strength of each one has been weighted, as well as linguistic and syntactic metrics using the WordNet dictionary. Finally, a threshold has been implemented in the final result in order to accurately and efficiently detect the semantic annotations for BPMN activity labels. In our implementation, the existence of suitable domain ontologies in SeMFIS database is uncertain. Whenever the concepts of one or more ontologies, which is created or inserted in the semfis database, match with the activity labels of a business process model, the system automatically suggests this ontology as candidate for annotating, using the same procedure of similarity measures followed for suggested semantic annotations.

Apart from adding semi-automatically suggested semantic annotations on existing BPMN process models, we have also extended the SeMFIS tool to export the existing ontologies in its database, as well as the annotated BPMN models which are created in it, using an RDF/OWL format so as to make them machine-accessible. For this purpose, we have created a basic BPMN ontology which encodes the classification of the most basic elements of BPMN, together with the most useful attributes and conditions describing how the elements can be combined to obtain a BPMN process model. Finally, we exploited the basic BPMN ontology to querying, using SPARQL language, the existing semantically annotated BPMN models, mainly for reusing them or fragments of them which will decrease the effort and time required for modeling any new processes.

In our future work, we will validate our approach employing more users to implement it and larger repositories. Our approach was performed for processes modeled in BPMN 2.0 language. Further generalization of our strategy for other process modeling notations is an upcoming work, too.

Appendix A

Appendix

A.1 Requirements

In order to be able to use the extended SeMFIS tool, with its new functionalities, the user have to install the following software:

- WordNet 3.0 for Windows English dictionary
- Virtuoso - Open Link Server

WordNet Installation

The WordNet 3.0 for Windows English dictionary is available for downloading at: http://wordnet.prenceton.edu/wordnet/publications/ and the user have to install it under the path: "C:\WordNet-3.0\dict".

Virtuoso Installation

The user have to download Virtuoso - Open Link pre-built binaries from: https://sourceforge.net/projects/virtuoso/files/virtuoso/6.1.8/virtuoso-opensource-x64-20131211.zip/download for Windows 64 bit and https://sourceforge.net/projects/virtuoso/files/virtuoso/6.1.8/virtuoso-opensource-x86-20131211.zip/download for Windows 32 bit.

The pre-built binaries of Virtuoso for windows require the Microsoft Visual C++ 2010 redistributable package be installed. If the user has not installed it earlier, he/ she can download it from the following locations: http://www.microsoft.com/download/en/details.aspx?id=5555 for 32-bit windows and http://www.microsoft.com/download/en/details.aspx?id=14632 for 64-bit windows.

In order to install the pre-binaries of Virtuoso the following instructions must be followed:

- Unzip in a folder (e.g. C://virtuoso-opensource/). This will create a directory virtuoso-opensource, containing the following subfolders:

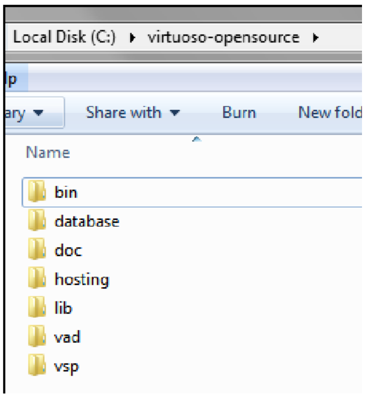


Figure A.1.1: Virtuoso subfolders

- Start the system control panel
    - Right click "My Computer" → "Properties" → "Advanced System Setting".
- Click to "Advanced" → "Environment Variables", create a new system environment variable called VIRTUOS_HOME and put as value the folder path of virtuoso (e.g. C://virtuoso-opensource/).
- Locate the PATH system environment variable and click to edit it.
- Add the string below to the end of the existing path value:
    - ;%VIRTUOSO_HOME%/bin;%VIRTUOSO_HOME%/lib
- IMPORTANT: do not over write the existing path value. Doing so will disrupt all use of your Windows environment.
- Click ok or exit buttons until you have fully exited the System Control Panel.
- Download the following php.ini file and manually place it in the "database" directory of Virtuoso: ftp://download.openlinksw.com/support/vos/php.ini
- Run a command line window as administrator: e.g. search "cmd", right click on "cmd.exe", click "run as administrator".
- Verify your virtuoso binary by typing in the command: "virtuoso-t-?"
- Go to the folder "database": e.g. run the command cd %VIRTUOSO_HOME%/database
- Create a new Windows Service with the following command: virtuoso-t +service create +instance MyVirtuosoInst +configfile virtuoso.ini
- A new service with the name "Virtuoso_MyVirtuosoInst" has been created.
- Once created, you can manage the Virtuoso server through the standard windows services manager (start → control panel → administrative tools → services or component services). It will be listed as OpenLink Virtuoso Server [Instance name].
- Start the Virtuoso Server: e.g. virtuoso-t +instance MyVirtuosoInst +service start.

Limitations

The Ontologies which are created, inserted or updated on SeMFIS database should not contain classes, properties or instances with the word "check" in its name. If there is one, it must be changed with another synonym word.

The BPMN process model names should not contain underlines (e.g. on_line_shop) otherwise there will be problem with automation of the annotated BPMN model names. It is better to separate words that make up a BPMN model name with the first letter of each word to be capitalized (e.g. onLineShop).

A.2 SPARQL Queries Syntax

The syntax of the seven standardized SPARQL queries is:

Table A.2.1: SPARQL Queries Syntax

Java SPARQL Queries syntax	**SPARQL Queries syntax**
switch(sparqlQueries){	

case "**1**. Give all the activities which have semantic annotation with the class ... as well as the process which belongs in.":
myPanel.add(new JLabel("Give all the activities which have semantic annotation with the class\n"));
String inputField = (String)JOptionPane.showInputDialog(null, myPanel, "input field", JOptionPane.PLAIN_MESSAGE);
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?activity ?process\n"
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"{?activity a basicBPMN_Ontology:Task.}\n" +
"UNION\n" +
"{?activity a basicBPMN_Ontology:Sub-Process.}\n"
"?activity basicBPMN_Ontology:has_SemanticAnnotation basicBPMN_Ontology:"+inputField.replace(" ", "_")+".\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.\n" +
"}";
break;
	Query 1. "*Give all the activities which have semantic annotation with the class ... as well as the process which belongs in*."
	
	Syntax:
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?activity ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Task.}
	UNION
	{?activity a basicBPMN_Ontology:Sub-Process.}
	?activity basicBPMN_Ontology:has_SemanticAnnotation basicBPMN_Ontology:inputField.replace(" ", "_")
	?activity basicBPMN_Ontology:isBaseElementOf ?process
	}

case "**2**. Give all distinct activities inside a subprocess and the subprocess.":
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n"
"\n" +
"select DISTINCT ?activity ?subprocess\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"{?activity a basicBPMN_Ontology:Task.}\n" +
"UNION\n" +
	Query 2. "*Give all distinct activities inside a subprocess and the subprocess.*"
	
	Syntax:
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT DISTINCT ?activity ?subprocess
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Task.}

"{?activity a basicBPMN_Ontology:Sub-Process.}\n" + "?sub a basicBPMN_Ontology:Sub-Process.\n" + "?sub basicBPMN_Ontology:has_SubprocessRef ?subprocess.\n" + "?activity basicBPMN_Ontology:isBaseElementOf ?subprocess.\n" + "}"; break;	UNION {?activity a basicBPMN_Ontology:Sub-Process.} ?sub a basicBPMN_Ontology:Sub-Process. ?sub basicBPMN_Ontology:has_SubprocessRef ?subprocess. ?activity basicBPMN_Ontology:isBaseElementOf ?subprocess. }

case "**3**. Give all the distinct activities of a process which consist subprocess of another process.":
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select DISTINCT ?activity ?subprocess\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"{?activity a basicBPMN_Ontology:Task.}\n" +
"UNION\n" +
"{?activity a basicBPMN_Ontology:Sub-Process.}\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.\n" +
"?sub basicBPMN_Ontology:isBaseElementOf ?pr2.\n" +
"?sub basicBPMN_Ontology:has_SubprocessRef ?subprocess.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?subprocess.\n" +
"}";
break;
	Query 3. " *Give all the distinct activities of a process which consist subprocess of another process.*"
	
	Syntax:
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT DISTINCT ?activity ?subprocess
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Task.}
	UNION
	{?activity a basicBPMN_Ontology:Sub-Process.}
	?activity basicBPMN_Ontology:isBaseElementOf ?process.
	?sub basicBPMN_Ontology:isBaseElementOf ?pr2.
	?sub basicBPMN_Ontology:has_SubprocessRef ?subprocess.
	?activity basicBPMN_Ontology:isBaseElementOf ?subprocess.
	}

case "**4**. Get all activities which are connected with the activity ... with the relation ...":
String[] relationOptions = {"Associations", "Data_Associations", "Message_Flows", "Sequence_Flows"};
JTextField inputTextField = new JTextField(20);
myPanel.add(new JLabel("Get all activities which are connected with the activity\n"));
myPanel.add(inputTextField);
myPanel.add(Box.createVerticalStrut(60)); // a spacer
myPanel.add(new JLabel("with the relation:\n"));
inputField = (String)JOptionPane.showInputDialog(null, myPanel, "input field", JOptionPane.PLAIN_MESSAGE, null, relationOptions, relationOptions[0]);
if(inputField.contentEquals("Associations")) {
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?activity ?process\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"{?activity a basicBPMN_Ontology:Text_Annotation.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" +
"UNION\n" +
	Query 4. " *Get all activities which are connected with the activity ... with the relation ...*"
	
	Syntax:
	
	Associations
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?activity ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Text_Annotation.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	UNION
	{?activity a basicBPMN_Ontology:Group.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	?x basicBPMN_Ontology:has_AssociationSourceRef basicBPMN_Ontology:inputTextField.getText().replace(" ", "_").
	?x basicBPMN_Ontology:has_AssociationTargetRef ?activity.
	}
	
	Data_Associations
	PREFIX basicBPMN_Ontology:

"{?activity a basicBPMN_Ontology:Group.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" +
"?x basicBPMN_Ontology:has_AssociationSourceRef basicBPMN_Ontology:"+inputTextField.getText().replace(" ", "_")+".\n" +
"?x basicBPMN_Ontology:has_AssociationTargetRef ?activity.\n" +
"}";
} else if(inputField.contentEquals("Data_Associations")) {
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?activity ?process\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"?activity a basicBPMN_Ontology:Data_Object.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.\n" +
"?x basicBPMN_Ontology:has_DataAssociationSourceRef basicBPMN_Ontology:"+inputTextField.getText().replace(" ", "_")+".\n" +
"?x basicBPMN_Ontology:has_DataAssociationTargetRef ?activity.\n" +
"}";
} else if(inputField.contains("Message_Flows")) {
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?activity ?process\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"{?activity a basicBPMN_Ontology:Task.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" +
 "UNION\n" +
"{?activity a basicBPMN_Ontology:Sub-Process.\n" +
"?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" +
"?x basicBPMN_Ontology:has_MessageSourceRef basicBPMN_Ontology:"+inputTextField.getText().replace(" ", "_")+".\n" +
"OPTIONAL{?x basicBPMN_Ontology:has_MessageTargetRef ?activity.}\n" +
"?x basicBPMN_Ontology:has_MessageTargetRef ?y.\n" +
"?z basicBPMN_Ontology:has_SequenceSourceRef ?y.\n" +
"?z basicBPMN_Ontology:has_SequenceTargetRef ?activity.\n" +
"}";
}else if (inputField.contains("Sequence_Flows")) {
sparqlQueryString = "PREFIX basicBPMN_Ontology:
	<http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?activity ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	?activity a basicBPMN_Ontology:Data_Object.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.
	?x basicBPMN_Ontology:has_DataAssociationSourceRef basicBPMN_Ontology:inputTextField.getText().replace(" ", "_").
	?x basicBPMN_Ontology:has_DataAssociationTargetRef ?activity.
	}
	
	Message_Flows
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?activity ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Task.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	UNION
	{?activity a basicBPMN_Ontology:Sub-Process.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	?x basicBPMN_Ontology:has_MessageSourceRef basicBPMN_Ontology:inputTextField.getText().replace(" ", "_").
	OPTIONAL{?x basicBPMN_Ontology:has_MessageTargetRef ?activity.}
	?x basicBPMN_Ontology:has_MessageTargetRef ?y.
	?z basicBPMN_Ontology:has_SequenceSourceRef ?y.
	?z basicBPMN_Ontology:has_SequenceTargetRef ?activity.
	}
	
	Sequence_Flows
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?activity ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	{?activity a basicBPMN_Ontology:Task.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	UNION
	{?activity a basicBPMN_Ontology:Sub-Process.
	?activity basicBPMN_Ontology:isBaseElementOf ?process.}
	?x basicBPMN_Ontology:has_SequenceSourceRef basicBPMN_Ontology:inputTextField.getText().replace

<http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" + "\n" + "select ?activity ?process\n" + "FROM <http://localhost:8890/BPMNontology> \n"+ "where {\n" + "{?activity a basicBPMN_Ontology:Task.\n" + "?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" + "UNION\n" + "{?activity a basicBPMN_Ontology:Sub-Process.\n" + "?activity basicBPMN_Ontology:isBaseElementOf ?process.}\n" + "?x basicBPMN_Ontology:has_SequenceSourceRef basicBPMN_Ontology:"+inputTextField.getText().replace(" ", "_")+".\n" + "{?x basicBPMN_Ontology:has_SequenceTargetRef ?activity.}\n"+ "UNION\n" + "{?x basicBPMN_Ontology:has_SequenceTargetRef ?y.\n" + "?z basicBPMN_Ontology:has_SequenceSourceRef ?y.\n" + "?z basicBPMN_Ontology:has_SequenceTargetRef ?activity.}\n" + "}"; } break;	(" ", "_"). {?x basicBPMN_Ontology:has_SequenceTargetRef ?activity.} UNION {?x basicBPMN_Ontology:has_SequenceTargetRef ?y. ?z basicBPMN_Ontology:has_SequenceSourceRef ?y. ?z basicBPMN_Ontology:has_SequenceTargetRef ?activity.} }

case "**5**. How many BPMN processes are related with ...":
myPanel.add(new JLabel("How many BPMN processes are related with\n"));
inputField = (String)JOptionPane.showInputDialog(null, myPanel, "input field", JOptionPane.PLAIN_MESSAGE);
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?process\n"
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"?process a basicBPMN_Ontology:BPMN_Process.\n" +
"FILTER regex(?process, \".*"+inputField+"*.\", \"i\")\n" +
"}";
break;
	Query 5. "*How many BPMN processes are related with ....*"
	
	Syntax:
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?process
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	?process a basicBPMN_Ontology:BPMN_Process.
	FILTER regex(?process, \".*"inputField"*.\", \"i\")
	}

case "**6**. Give all the models that include the subprocess: ...": myPanel.add(new JLabel("Give all the models that include the subprocess\n")); inputField = (String)JOptionPane.showInputDialog(null, myPanel, "input field", JOptionPane.PLAIN_MESSAGE); sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" + "\n" +	**Query 6**. " *Give all the models that include the subprocess: ....*" **Syntax:** PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/> SELECT ?process FROM <http://localhost:8890/BPMNontology> WHERE { ?sub a basicBPMN_Ontology:Sub-Process. ?sub basicBPMN_Ontology:isBaseElementOf ?process.

"select ?process\n" + "FROM <http://localhost:8890/BPMNontology> \n"+ "where {\n" + "?sub a basicBPMN_Ontology:Sub-Process.\n" + "?sub basicBPMN_Ontology:isBaseElementOf ?process.\n" + "?sub basicBPMN_Ontology:has_SubprocessRef basicBPMN_Ontology:"+inputField+".\n" + "}"; break;	?sub basicBPMN_Ontology:has_SubprocessRef basicBPMN_Ontology:inputField }

case "**7**. Give all the common subprocesses of a whole of models.":
sparqlQueryString = "PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>\n" +
"\n" +
"select ?subprocess\n" +
"FROM <http://localhost:8890/BPMNontology> \n"+
"where {\n" +
"?sub1 a basicBPMN_Ontology:Sub-Process.\n" +
"?sub1 basicBPMN_Ontology:has_SubprocessRef ?subprocess.\n" +
"?sub1 basicBPMN_Ontology:isBaseElementOf ?pr1.\n" +
"?sub2 a basicBPMN_Ontology:Sub-Process.\n" +
"?sub2 basicBPMN_Ontology:has_SubprocessRef ?subprocess.\n" +
"?sub2 basicBPMN_Ontology:isBaseElementOf ?pr2.\n" +
"FILTER (?pr1 != ?pr2)\n" +
"}";
break;
	Query 7. "*Give all the common subprocesses of a whole of models.*"
	
	Syntax:
	PREFIX basicBPMN_Ontology: <http://www.semanticweb.org/ioanna/ontologies/2016/3/basicBPMN_Ontology#/>
	SELECT ?subprocess
	FROM <http://localhost:8890/BPMNontology>
	WHERE {
	?sub1 a basicBPMN_Ontology:Sub-Process.
	?sub1 basicBPMN_Ontology:has_SubprocessRef ?subprocess.
	?sub1 basicBPMN_Ontology:isBaseElementOf ?pr1.
	?sub2 a basicBPMN_Ontology:Sub-Process.
	?sub2 basicBPMN_Ontology:has_SubprocessRef ?subprocess.
	?sub2 basicBPMN_Ontology:isBaseElementOf ?pr2.
	FILTER (?pr1 != ?pr2)
	}

Bibliography

[1] Branimir Wetzstein, Zhilei Ma, Agata Filipowska, Monika Kaczmarek, Sami Bhiri, Silvestre Losada, Jose-Manuel Lopez-Cobo, Laurent Cicurel, "Semantic business process management: A lifecycle based requirements analysis", In Proc. of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM 2007) at the 4th European Semantic Web Conference (ESWC 2007), pages 1–11. CEUR WS, 2007.

[2] Born, M., Dorr, F., Weber, I., "User-friendly Semantic Annotation in Business Process Modeling", In: Web Information Systems Engineering (WISE) Workshops, 2007, p. 260-271.

[3] C. Di Francescomarino, C. Ghidini, M. Rospocher, L. Serafini, P. Tonella, "Semantically-Aided Business Process Modeling", in: ISWC 2009, Vol. 5823 of LNCS, Springer, Berlin, Heidelberg, 2009, pp. 114–129

[4] C. Di Francescomarino, P. Tonella, "Supporting Ontology-Based Semantic Annotation of Business Process with Automated Suggestions", in: Enterprise, Business-Process and Information Systems Modeling, volume 29 of Lecture Notes in Business Information Processing, Springer, 2009, pp. 211–223.

[5] Markovic, I., Pereira, A.C., "Towards a Formal Framework for Reuse in Business Process Modeling", In Workshop on Advances in Semantics for Web services (semantics4ws), in conjunction with BPM '07, Brisbane, Australia, September 2007.

[6] Hepp, Martin; Leymann, Frank; Bussler, Chris; Domingue, John; Wahler, Alexander and Fensel, Dieter (2005), "Semantic business process management: a vision towards using semantic web services for business process management", In: IEEE International Conference on e Business Engineering, 18-20 Oct 2005, Beijing, China.

[7] M. Ehrig, A. Koschmider, A. Oberweis, "Measuring similarity between semantic business process models", in APCCM ’07: Proceedings of the fourth Asia-Pacific conference on Comceptual modelling. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007, pp. 71–80.

[8] M. Dimitrov, A. Simov, S. Stein, and M. Konstantinov, "A bpmo based semantic business process modelling environment", In Proceedings of the

Workshop on Semantic Business Process and Product Lifecycle Management at the ESWC, volume 251 of CEUR-WS, 2007.

[9] Hepp, M., Roman, D. (2007), "An ontology framework for semantic business process management", Proceedings of Wirtschaftsinformatik 2007, Karlsruhe, Germany, pp. 1-18.

[10] Vazquez, B., Martinez, A., Perini, A., Estrada, H., Morandini, M., "Enriching Organizational Models through Semantic Annotation", In Proceedings of the Iberoamerican Conference on Electronics Engineering and Computer Science, 2013.

[11] Francescomarino, C. Di, Rospocher, M., Ghidini, C., & Valerio, A., "The Role of Semantic Annotations in Business Process Modeling", In Proceedings of the 18th International Conference on Enterprise Distributed Object Computing Conference (EDOC ’14), Ulm, Germany, September 1-5, 2014 (pp. 181–189). IEEE Computer Society Press.

[12] van der Aalst WMP (2013), "Business Process Management: A Comprehensive Survey", ISRN softw Eng 2013.

[13] B. List, B. Korherr, "An evaluation of conceptual business process modelling languages", In H. Haddad, editor, Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006, pages 1532–1539. ACM, 2006.

[14] O. Thomas and M. Fellmann, "Semantic epc: Enhancing process modeling using ontology languages", In Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM), pages 64–75, June 2007.

[15] Rospocher, M., Ghidini, C., Serafini, L., "An Ontology for the Business Process Modelling Notation", In: Formal Ontology in Information Systems - Proceedings of the 8th FOIS. vol. 267, pp. 133–146. IOS Press (2014).

[16] White, S.A., "Introduction to BPMN", Business Process Trends, July 2004.

[17] Malik, S., Bajwa I.S., "Back to Origin: Transformation of Business Process Models to Business Rules", in Business Process Management Workshops, Springer, 2013, pages 611-622.

[18] G. Della Penna, R. Del Sordo, B. Intrigila, N. Mezzopera, M. T. Pazienza, "A Lightweight Formalism for the Integration of BPMN Models with Domain Ontologies", 2013.

[19] Weissgerber, A., "Semantically-enriched business process modeling and management", Ph.D. thesis, Universitat des Saarlandes, Germany (2011).

[20] Object Management Group (OMG), "Business Process Model and Notation (BPMN)", Version 2.0, January 2011, OMG Document Number: formal/2011-01-03. This file was downloaded from: http://www.omg.org/spec/BPMN/2.0/PDF

[21] Dijkman, Remco M., Dumas, Marlon, & Ouyang, Chun (2008) "Semantics and analysis of business process models in BPMN", Information and Software Technology, 50(12), pp. 1281-1294. This file was downloaded from: http://eprints.qut.edu.au/7115/

[22] Antoniou, G., Van Harmelen, F. "A Semantic Web Primer", p.10, 61-88, 109-143, 2004, Massachusetts Institute of Technology.

[23] Fensel, D. " Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce", February 2000, Springer-Verlag.

[24] Daskalaki, Ev. "Development and Experimental Evaluation of an Ontology to Ontology Schema & Instance Matching System", Master Thesis – University of Crete, Department of Computer Science", October 2011.

[25] Oren, E., Hinnerk Möller, K., Scerri, S., Handschuh, S., Sintek, M.(2006). What are Semantic Annotations?. Technical report, Digital Enterprise Research Institute (DERI), National University of Ireland, Galway.

[26] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press. Available at: http://wordnet.prenceton.edu/wordnet/publications/

[27] Activiti at: http://activiti.org/

[28] Bpmn2-modeler at: https://www.eclipse.org/bpmn2-modeler/

[29] Camunda at: https://camunda.org

[30] Fill, H.-G., "SeMFIS: a Flexible Engineering Platform for Semantic Annotations of Conceptual Models", Semantic Web Journal, IOS press, 2015.

[31] Adonis at: https://uk.boc-group.com/adonis/

[32] K. Grolinger, M.A.M. Capretz, A. Cunha, S. Tazi, "Integration of Business Process Modeling and Web Services: A Survey", Service- Oriented Computing and Applications (SOCA), 2013. The final publication is available at link.springer.com: http://link.springer.com/article/10.1007%2Fs11761-013-0138-2

[33] Cabral, L., Norton, B., Domingue, J., (2009) "The business process modelling ontology", In: 4th International Workshop on Semantic Business Process

Management (SBPM 2009), Workshop at ESWC 2009, 1 June 2009, Crete, Greece.

[34] Lin, Y., Strasunskas, D., Hakkarainen, S., Krogstie, J., Solvberg, A., (2006) "Semantic annotation framework to manage semantic heterogeneity of process models", Advanced information systems engineering. Springer, Berlin, Heidelberg, pp. 433-446.

[35] Lin, Y., Ding, H., (2005) "Ontology-based semantic annotation for semantic interoperability of process models", In: Proc Int Conf Comput Intell Model Control Automat.

[36] Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P., "Reasoning on Semantically Annotated Processes", In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 132–146. Springer, Heidelberg (2008).

[37] Ciuciu, I. G., Tang, Y. and Meersman, R. (2011), "Towards Retrieving and Recommending Security Annotations for Business Process Models Using an Ontology-based Data Matching Strategy", in Proc. of the first international symposium on data-driven process discovery and anlysis (SIMPDA'11), IFIP working group 2.6 and 2.12, ISBN 978- 88-903120-2-1, vol. 1, pp. 71-81, Campion d'italia, Italy, June 29th ~ July 1st, 2011.

[38] JWNL 1.4-rc3, a java WordNet library available at: http://sourceforge.net/projects/jwordnet/files/

[39] JWNL javadoc at: https://web.stanford.edu/class/cs276a/projects/docs/jwnl/javadoc/

[40] Short JWNL tutorial at: http://blog.roland-kluge.de/?p=430

[41] JWI 2.3.3, a Small java WordNet library available at: http://projects.csail.mit.edu/jwi/

[42] N-gram Algorithm: https://en.wikipedia.org/wiki/N-gram

[43] Bigram Algorithm: https://en.wikipedia.org/wiki/Bigram

[44] Dice's Coefficient formula: https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient

[45] Trigram Algorithm: https://en.wikipedia.org/wiki/Trigram

[46] Levenshtein Distance Algorithm: https://en.wikipedia.org/wiki/Levenshtein_distance

[47] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records", In Proceedings of the workshop on Data

Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining (KDD), 2003.

[48] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string distance metrics for Name - Matching Tasks", In Proceedings of the IJCAI-2003 Workshop on Information, 2003.

[49] Jaro - Winkler Distance Algorithm: https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

[50] Jaccard Similarity Algorithm: https://en.wikipedia.org/wiki/Jaccard_index

[51] Soundex Algorithm: https://en.wikipedia.org/wiki/Soundex

[52] Soundex Algorithm: http://creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm

[53] Char Frequency Similarity: http://rosettacode.org/wiki/Most_frequent_k_chars_distance#Java

[54] Humm, G. B., Fengel, J., (2012): "Semantics-based Business Process Model Similarity", In: Abramowicz, W., Krikscieuniene, D., Sakalauskas, V. (eds.): Proceedings of the 15th International Conference on Business Information Systems (BIS 2012), LNBIP 117. Springer Berlin Heidelberg, pp. 36–47

[55] Eclipse - luna tool at https://eclipse.org/luna/

[56] Becker, M., Laue, R., "Analysing differences between business process similarity measures", In: 1st International Workshop on Process Model Collections, pp. 39–49 (2011).

[57] Embley, W.,D., Jackman, D, Xu, L., "Multifaceted exploitation of metadata for attribute match discovery in information integration", In: Proc Int Workshop on Information Integration on the Web, pp. 110–117, 2001.

[58] Sparql Query Language for RDF: https://www.w3.org/TR/rdf-sparql-query/#basicpatterns

[59] OpenLink Virtuoso: https://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso

[60] Smith, F., & Proietti, M. (2013), "Rule-based behavioral reasoning on semantic business processes", In Proc. of the 5th International Conference on Agents and Artificial Intelligence, SciTePress Digital Library.

[61] Born, M., Hoffmann, J., Kaczmarek, T., Kowalkiewicz, M., Markovic, I., Scicluna, J., Weber, I., Zhou, X., "Semantic annotation and composition of business processes with Maestro", In: European Semantic Web Conference (ESWC) Demo Track, June 2008

[62] RDF Description at: https://www.w3.org/RDF/

[63] RDF-Schema Description at: https://www.w3.org/TR/rdf-schema/

[64] RDF-Schema Description at: https://en.wikipedia.org/wiki/RDF_Schema

[65] Slides of the course CS561, Computer Science Department (CSD), University of Crete (UOC), Yannis Tzitzikas, Spring 2015.

[66] Slides of the course CS561, Computer Science Department (CSD), University of Crete (UOC), Yannis Tzitzikas, Spring 2015.

[67] OWL at https://www.w3.org/TR/owl-features/