Semantically-Aided Business Process Modeling

Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher,
Luciano Serafini, and Paolo Tonella

FBK-irst, Via Sommarive 18 Povo, I-38123, Trento, Italy
{dfmchiara,ghidini,rospocher,serafini,tonella}@fbk.eu

Abstract. Enriching business process models with semantic annotations taken from an ontology has become a crucial necessity both in service provisioning, integration and composition, and in business processes management. In our work we represent semantically annotated business processes as part of an OWL knowledge base that formalises the business process structure, the business domain, and a set of criteria describing correct semantic annotations. In this paper we show how Semantic Web representation and reasoning techniques can be effectively applied to formalise, and automatically verify, sets of constraints on Business Process Diagrams that involve both knowledge about the domain and the process structure. We also present a tool for the automated transformation of an annotated Business Process Diagram into an OWL ontology. The use of the semantic web techniques and tool presented in the paper results in a novel support for the management of business processes in the phase of process modeling, whose feasibility and usefulness will be illustrated by means of a concrete example.

1 Introduction

Semantic Business Process Management [11,6] has the main objective of improving the level of automation in the specification, implementation, execution, and monitoring of business processes by extending business process management tools with the most significant results from the area of semantic web. When the focus is on process modeling, i.e. the activity of specification of business processes at an abstract level (descriptive and non executable), annotating process descriptions with labels taken from a set of domain ontologies provides additional support to the business analysis (see e.g. [16]).

A crucial step in process modeling is the creation of valid diagrams, which not only comply with the basic requirements of the process semantics, but also satisfy properties that take into account the domain specific semantics of the labels of the different process elements. For instance, an important requirement for a valid on-line shopping process should be the fact that *the activity of providing personal data is always preceded by an activity of reading the policy of the organization*. As the notion of semantically annotated processes becomes more and more popular, and business experts start to annotate elements of their processes with semantic objects taken from a domain ontology, there is an increasing potential to use Semantic Web technology to support business experts in their modeling activities, including the modeling of valid diagrams which satisfy semantically enriched and domain specific constraints. A clear demonstration of this, is the stream of recent work on the introduction and usage of formal semantics to support Business Process Management [19,20,13,2,5,18,15].

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 114–129, 2009.

Analyzing this stream of work we can roughly divide the different approaches into two groups: (i) those adding semantics to specify the *dynamic* behavior exhibited by a business process, and (ii) those adding semantics to specify the meaning of the entities of a business process in order to improve the automation of business process management. In this paper we place ourselves in the second group and we focus on the usage of Semantic Web technology to specify and verify *structural* constrains, that is, constraints that descend from structural requirements which refer to *descriptive* properties of the annotated process diagram and not to its execution. We focus on structural requirements for two fundamental reasons: first, structural requirements complement behavioral properties, as they can be used to express properties of the process which cannot be detected by observing the execution of a process. Second, structural requirements provide an important class of expressions whose satisfiability can be directly verified with existing Description Logic (DL) reasoners.

Thus, the purpose of this paper is to propose a concrete formalization of typical classes of structural requirements over annotated BPMN processes [4], and to show how DL reasoners can be used to provide the verification services to support modeling activities. More in detail: we first recall how to represent semantically annotated BPMN processes within an OWL-DL Business Process Knowledge Base (BPKB), firstly introduced in [7]; with respect to [7], we extend the structure of BPKB to incorporate constraints used to formalize structural requirements (Section 3); then we provide concrete examples of how to encode typical classes of structural requirements in BPKD (section 4); finally, we show how to automatically translate an annotated BPMN process into a set of assertions of the Business Process Knowledge Base and we evaluate the usage of Description Logic reasoners to validate structural requirements (Section 5). An example, introduced in Section 2, is used throughout the paper to illustrate the proposed formalism, while a comparison with related approaches is contained in Section 6.

2 A Motivating Example

In this section, we describe a portion of an on-line shopping process which we use throughout the paper as a motivating and explanatory example. The annotated process we refer to is illustrated in Figure 1. The Business Process Modeling Notation (BPMN) [17] is the (graphical) language used to draw Business Process Diagrams (BPD). In our semantic variant of BPMN we allow for the annotation of objects of a BPD with concept descriptions taken from domain ontologies, i.e. shared formalizations of a specific domain. Annotations are preceded in a BPD by the "@" symbol. Due to space limitations we consider here only the initial steps of the on-line shopping process (e.g. the product presentation and selection and the customer authentication), leaving out the last phases, e.g. the checkout.

The realization of the on-line shopping process depicted in Figure 1 can involve a team of business experts, who may wish to impose requirements (constraints) on the process itself. These requirements could cover different aspects of the process, ranging from the correct annotation of business processes to security issues, as in the following examples:

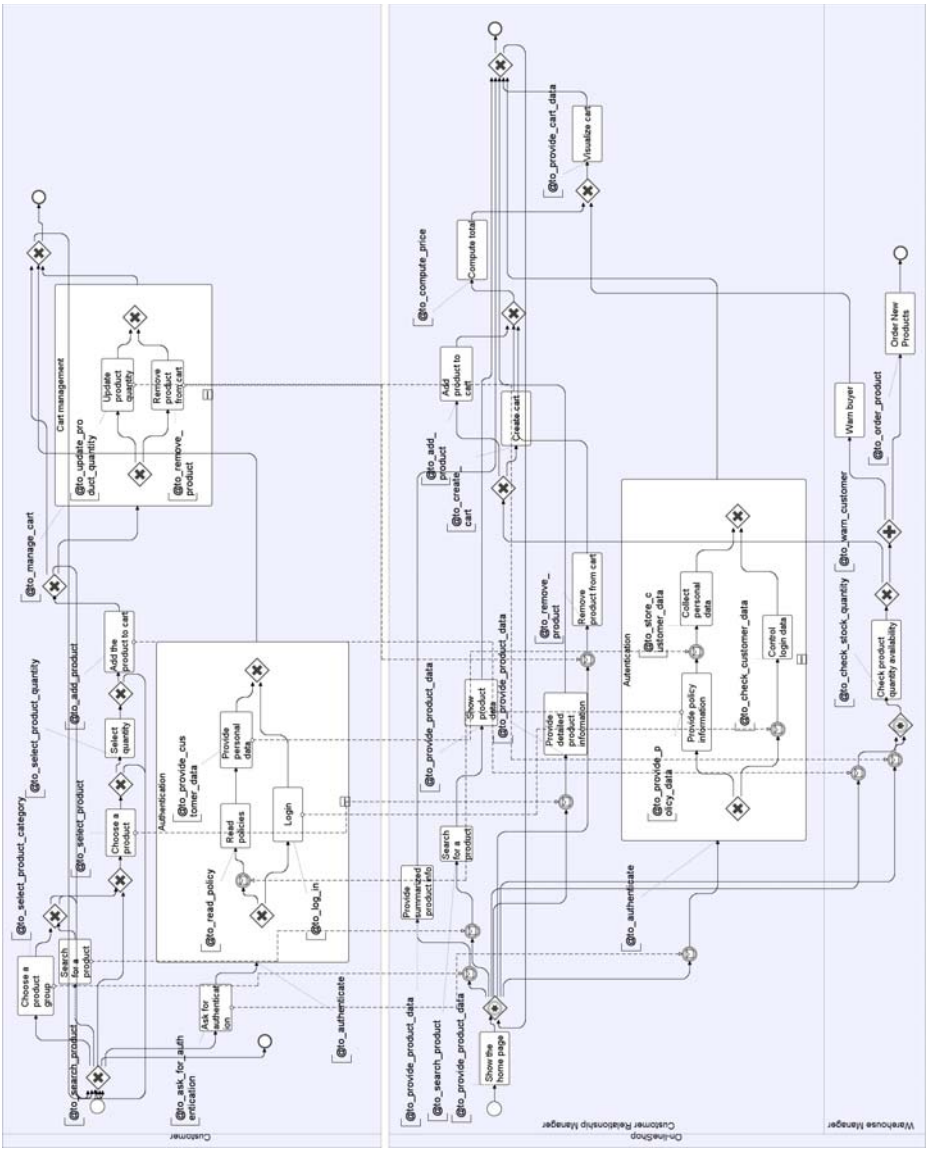

Fig. 1. A portion of the On-line shopping business process diagram

– issues related to the semantic annotation:
  (a) *"to_manage" is a complex action and can be used only to annotate BPMN sub-processes (and not atomic activities).*
– privacy issues:
  (b) *the activity of providing personal data is always preceded by an activity of reading the policy of the organization*;

(c) *the activity of reading the policy of the organization is activated by an event generated from an activity of providing these policies to the customer itself*;

– security issues:

(d) *the customer pool must contain an authentication sub-process which, in turn, contains a log-in activity and an insertion of personal data activity*;

– general issues :

(e) *in the on-line shopping process there must be a "Customer" pool and an "On-line shop" pool*;

(f) *inclusive gateway cannot be used in the on-line shopping process (to force all the alternative actions to be mutually exclusive)*;

(g) *each gateway must have at most 2 outgoing gates (to keep the process simple)*;

(h) *each pool must contain a single authentication activity / sub-process (to ease maintenance)*;

(i) *the activity of managing a shopping cart is a sub-process which contains an activity of removing products from the cart*.

All these constraints are examples of structural requirements as they relate to the descriptive properties of the annotated process diagram and complement properties which may refer to the process execution. While some of the requirements listed above can bear some similarity with behavioral properties, it is important to note here that expressing them as structural requirements allows to give them a different meaning, as it is well known that the same process behavior can be obtained by completely different process diagrams. To make a simple example we could "rephrase" constraint (h) in the apparently equivalent *the execution paths of all pools must contain a single authentication activity*. Nevertheless while this requirement is satisfied by both diagram in Figure 2, requirement (h) is only satisfied by diagram 2(b), which is definitely easier to maintain if changes to the authentication sub-process are foreseen. Thus, structural requirements are the appropriate way to express static properties of the diagram, which may even not be discovered by analyzing the behavior of the process.

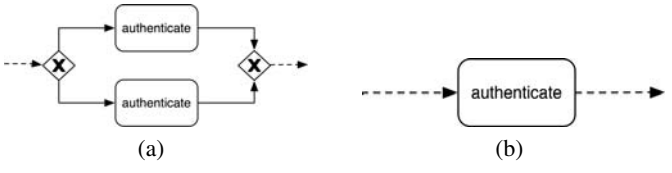

Fig. 2. Diagrams with equivalent behavior

3 Representing Semantically Annotated Processes

In order to represent semantically annotated BPDs, and to support automated reasoning on the requirements that can be expressed on them, we extend the notion of *Business*

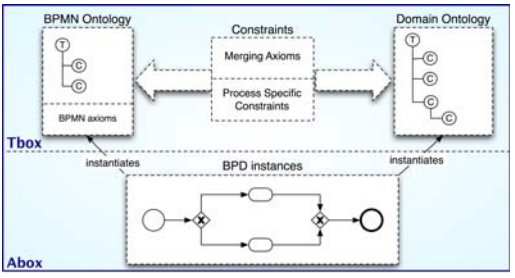

Fig. 3. The Business Processes Knowledge Base

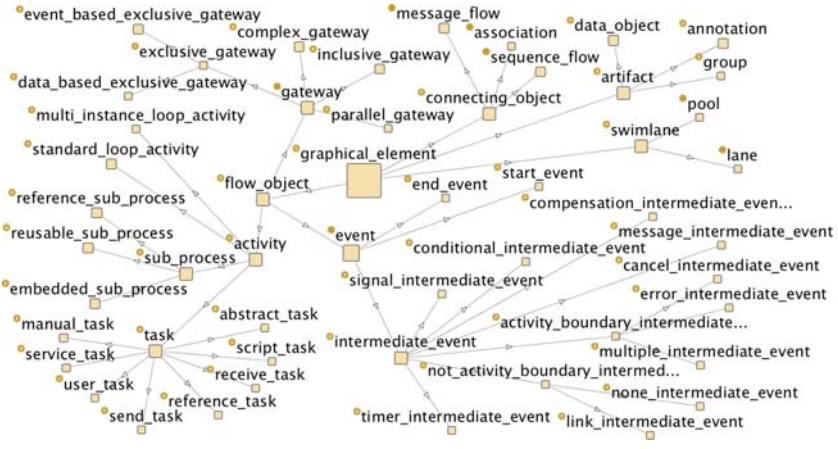

Fig. 4. The graphical elements of BPMNO

Processes Knowledge Base (BPKB), firstly introduced in [7] and schematized in Figure 3. The extension, described in details in Section 4, allows to define process specific constraints which are used to state specific structural requirements on the process to be modelled.

A BPKB is composed of four modules: a BPMN ontology, a domain ontology, a set of constraints and the BPD instances.

The BPMN Ontology. The BPMN ontology, hereafter called BPMNO, formalizes the structure of a BPD. It is a formalization of the BPMN standard as described in Annex B of [17], and consists of a set of axioms that describe the BPMN elements and the way in which they can be combined for the construction of BPDs. The taxonomy of the graphical elements of BPMNO is illustrated in Figure 4. The ontology has currently the expressiveness of $\mathcal{ALCHOIN}(\mathcal{D})$ and a detailed description is contained in [10]. We remark again that BPMNO provides a formalization of the structural part of BPDs, i.e. which are the basic elements of a BPD and how they are (can be) connected. BPMNO is not intended to model the dynamic behavior of BPDs (that is, how

the flow proceeds within a process). Ontology languages are not particularly suited to specify behavioral semantics. This part can be better modeled using formal languages for Workflow or Business Process Specification based on Petri Nets, as proposed in [13].

The Domain Ontology. The domain ontology component, hereafter called BDO, consists of a (set of) OWL ontology(es) that describes a specific business domain. It allows to give a precise semantics to the terms used to annotate business processes. The BDO can be an already existing business domain ontology (e.g. RosettaNet or similar standard business ontologies), a customization of an existing ontology, or an artefact developed on purpose. Top level ontologies such as DOLCE [8] can be included as "standard" components of the domain ontology and used to provide typical annotation patterns to the BPD objects.

The Constraints. Constraints are used to ensure that important semantic structural requirements of process elements are satisfied. We distinguish among two different kinds of constraints: **merging axioms** and **process specific constraints**. Merging axioms state the correspondence between the domain ontology and the BPMN ontology. They formalize the criteria for correct/incorrect semantic annotations. Process specific constraints are expressions used to state specific structural requirements that apply to the process under construction. Differently from merging axioms, these expressions can have many different forms to match a variety of different properties of the process.

The BPD Instances. The BPD instances (or BPD objects) contain the description of a set of annotated BPDs in terms of instances of the BPMN ontology and the domain ontology. Roughly speaking, the BPD instances obtained from an annotated BPD β are all the graphical objects g of β. The assertions on these elements can be divided into three groups: *BPM-type assertions*, *BPM-structural assertions* and *BPM-semantic assertions*. The first two groups of assertions involve concepts from BPMNO only, while the third group involves concepts from BDO only. BPM-type assertions are used to store informations on the type of graphical object g. Thus we represent the fact that g is an exclusive gateway with the BPM-type assertion data_based_exclusive_gateway(g). Analogously the assertion sequence_flow(s) states that the BPM object s is of type sequence_flow. BPM-structural assertions are used to store information on how the graphical objects are connected. Thus for every connecting object c of β that goes from a to b, we generate two structural assertions of the form SourceRef(c,a) and TargetRef(c,b). For instance, the assertion has_sequence_flow_source_ref(c,a) states that the sequence flow c originates from gateway a. Finally BPM-semantic assertions are used to represent annotation of graphical elements. For instance the assertion to_search_product(t) states that task t is an instance of concept to_search_product and is obtained from the semantic annotation to_search_product of the BPD in Figure 1.

We have implemented BPKB using the standard semantic web language OWL-DL based on Description Logics (DL) [1]. The terminological part (Tbox), which is the stable description of a given domain, is provided by the upper level modules of Figure 3. Instead, the changeable part, which corresponds to a specific process description, is provided in the form of assertional knowledge (Abox).

4 Specifying Structural Requirements

To ensure that important structural requirements are satisfied, we make use of constraints. We distinguish between two different kinds of constraints: **merging axioms** and **process specific constraints**.

4.1 Merging Axioms

Though belonging to different ontologies, concepts from the BMPN ontology and the domain ontology are not totally unrelated. Indeed, precise correspondences often exist which define criteria for correct / incorrect semantic annotations. Examples of these criteria, which may hold in many application domains, are:

A BPMN activity ***can be annotated only*** *with actions of the domain ontology (and not e.g., with objects).* (1)

A BPMN data-object ***cannot be annotated*** *with actions or events of the domain ontology (but e.g., with objects).* (2)

A BPMN Event ***can be annotated only*** *with events of the domain ontology (and not e.g., with objects).* (3)

An additional domain specific criterion, which refer to the particular business process or domain ontology at hand, is requirement (a) in Section 2.

To allow the business designer to specify the kind of positive and negative constraints described above, in [7] we have proposed the usage of four relations: "*annotatable only by*" ($\xrightarrow{\mathsf{AB}}$) and "*not annotatable by*" ($\xrightarrow{\mathsf{nAB}}$) from BPMNO concepts to BDO concepts, and the symmetrical "*annotates only*" ($\xrightarrow{\mathsf{A}}$) and "*cannot annotate*" ($\xrightarrow{\mathsf{nA}}$) from BDO concepts to BPMNO concepts. In the following table we report the intuitive meaning, and the formalization as DL axioms, of these four constructs. We use x to denote a concept of BPMNO and y to denote a concept of BDO.

Merging Axiom	Intuitive meaning	DL axiom[1]
$x \xrightarrow{\mathsf{AB}} y$	a BPMN element of type x can be annotated only with a concept equivalent or more specific than y	$x \sqsubseteq y$
$x \xrightarrow{\mathsf{nAB}} y$	a BPMN element of type x cannot be annotated with a concept equivalent or more specific than y	$x \sqsubseteq \neg y$
$y \xrightarrow{\mathsf{A}} x$	any concept equivalent or more specific than y can be used to denote only BPMN elements of type x	$y \sqsubseteq x$
$y \xrightarrow{\mathsf{nA}} x$	any concept equivalent or more specific than y can not be used to denote BPMN elements of type x	$y \sqsubseteq \neg x$

The formalization of the four constructs as DL axioms is the basis for the translation of the informal expressions such as (1)–(3) and (a) into a formal set of expressions, denoted with $\mathsf{Merging\_Axioms}(\mathsf{BPMNO}, \mathsf{BDO})$. Note that though the meaning of $x \xrightarrow{\mathsf{nAB}} y$ and $y \xrightarrow{\mathsf{nA}} x$ coincide, we provide both primitives as, depending on the case to be modeled, one may result more intuitive than the other.

Merging axioms can describe "domain independent" criteria, such as (1)–(3), and "domain specific" criteria, such as requirement (a). Domain independent criteria, may hold in many application domains, as they relate elements of BPMN, such as data-objects, activities or events to very general concepts, like the elements of a top-level ontology, e.g. DOLCE [8]. These kinds of constraints can be thought of as "default" criteria for correct / incorrect semantic annotations, and in this case DOLCE can be provided as a "default" component of the domain ontology in the workspace. The advantage of having these criteria already included in the BPKB is that in many situations it might be the case that the analysts, which are usually very focused on their application domain, forget to add them explicitly while they may tend to add more domain-specific constraints; note however that these "default" criteria could still be modified by the analysts to reflect the actual annotation criteria for the specific domain at hand.

To support the creation of merging axioms, we have implemented a first library of domain independent merging axioms between BPMN and DOLCE (see [9] for a detailed description). Based on this work, expression (1) can be represented with the merging axiom $\mathsf{activity} \xrightarrow{\mathsf{AB}} \mathsf{process}$ (identifying action with class process in DOLCE) which in turn is formally represented with the DL statement $\mathsf{BPMNO{:}activity} \sqsubseteq \mathsf{BDO{:}process}$, expression (2) can be represented with the merging axiom $\mathsf{data\_object} \xrightarrow{\mathsf{nAB}} \mathsf{perdurant}$ (where DOLCE class perdurant is a general class covering both processes and events) which in turn is represented with $\mathsf{BPMNO{:}data\_object} \sqsubseteq \neg\mathsf{BDO{:}perdurant}$, and similarly with the other expressions.

4.2 Process Specific Constraints

These constraints are expressions used to state specific properties that apply to the process under construction. Differently from merging axioms, these expressions can have many different forms to match a variety of different properties of the process. In this paper we focus on three types of process specific constraints that can be expressed over the Business Process Diagrams: (i) containment constraints (including existence constraints), (ii) enumeration constraints, and (iii) precedence constraints.

Containment Constraints. Containment constraints are of the form *X contains Y* or *X is contained in Y* and are used to represent the fact that the BPD or certain graphical elements contain other graphical elements. As such they can be used to express also informal statements of the form *exists X* and *non exists X*, which are rephrased in the containment constraint *diagram X contains Y* and *diagram X does not contain Y*. A simple containment constraint of the form *X contains Y* which can be expressed over the on-line shopping process is provided by requirement (i). A constraint of the form *exists X* is instead provided by requirements (e), while a constraint of the form *non exists X* is given by requirement (f).

Containment constraints can be encoded in Description Logics using specific BPMNO roles which formalise the containment relations existing between different BPD objects as described by specific attributes in [17]. Examples of these roles, used in DL to represent object properties and data properties, are:

- has_embedded_sub_process_sub_graphical_elements. This role corresponds to the GraphicalElement attribute of an Embedded Sub-Process, as described in [17], and represents all of the objects (e.g., Events, Activities, Gateways, and Artifacts) that are contained within the Embedded Sub-Process;
- has_pool_process_ref which corresponds to the ProcessRef attribute and is used to represent the process that is contained within a pool;
- has_process_graphical_element which corresponds to the GraphicalElements attribute of BPMN and identifies all of the objects that are contained within a process;
- has_business_process_diagram_pools which allows to relate a BPD with the pools it contains.

Using r to indicate one of the roles above, containment constraints are typically expressed as statements of the form $X \sqsubseteq \exists r.Y$ or $X \sqsubseteq \forall r.Y$ which use the basic existential and universal quantification constructs $\exists r.Y$ and $\forall r.Y$. Requirement (i) can therefore be formalized as follows, where we use has_embedded as a shorthand for has_embedded_sub_process_sub_graphical_elements for the sake of readability:

$$\text{BDO:to\_manage\_cart} \sqsubseteq \text{BPMN:embedded\_sub\_process} \tag{4}$$

$$\text{BDO:to\_manage\_cart} \sqsubseteq \exists \text{BPMN:has\_embedded.(BPMN:activity} \sqcap \text{BDO:to\_remove\_product)} \tag{5}$$

Similarly, requirement (e) can be encoded as follows:

$$\begin{aligned}&\text{BPMNO:business\_process\_diagram} \sqsubseteq \\ &\quad \exists \text{BPMNO:has\_business\_process\_diagram\_pools.BDO:customer} \sqcap \\ &\quad \exists \text{BPMNO:has\_business\_process\_diagram\_pools.BDO:on-line\_shop}\end{aligned} \tag{6}$$

while requirement (f) can be formalized by asserting:

$$\begin{aligned}&\text{BPMNO:process} \sqsubseteq \\ &\quad \forall \text{BPMNO:has\_process\_graphical\_element.} \neg \text{BPMNO:inclusive\_gateway}\end{aligned} \tag{7}$$

A more complex example of containment constraint is provided by requirement (d). The formalization of this constraint is omitted for lack of space.

Enumeration Constraints. Enumeration constraints further refine containment constraints by stating that *X contains (at least / at most / exactly) n objects of type X*. A simple example of enumeration constraint which concern a plain BPMN element is provided by requirements (g). An enumeration constraint which also involves semantically annotated objects is provided by requirement (h). Enumeration constraints can be encoded in Description Logics using the constructors of *number restriction* and *qualified number restriction* [1]. Number restrictions are written as $\geq nR$ (at-least restriction) and $\leq nR$ (at-most restriction), with n positive integer, while qualified number restrictions are written as $\geq nR.C$ and $\leq nR.C$. The difference between the two is that number restriction allows to write expressions such as, e.g., $\leq 3 hasChild$, which characterise the set of individuals who have at most 3 children, while qualified number restriction

allows to write expressions such as, e.g., $\leq 3 hasChild.Female$, which characterise the set of individuals who have at most 3 female children. At-least and at-most operators can be combined to obtain statement of the form $=nR$.

Thus, a formalization of requirements (g) can be provided by the DL statement:

$$\mathsf{BPMNO{:}gateway} \sqsubseteq (\leq 2)\mathsf{BPMNO{:}has\_gateway\_gate} \tag{8}$$

while a formalization of requirement (h) is given by:

$$\begin{aligned}\mathsf{BPMN{:}pool} \sqsubseteq \forall \mathsf{BPMN{:}has\_pool\_process\_ref}.\\ (=1)\mathsf{BPMN{:}has\_process\_graphical\_element.BDO{:}to\_authenticate}\end{aligned} \tag{9}$$

Precedence Constraints. Precedence constraints are used to represent the fact that certain graphical objects appear before others in the BPD. They can be of several forms. Significant examples are: *X is always preceded by Y* in all possible paths made of sequence flows and *X is once preceded by Y* in at least a path composition of sequence flows. Particular cases of these constraints are *X is always immediately preceded by Y* and *X is once immediately preceded by Y*. These constraints also require that *X* is a graphical object immediately preceded by *Y* by means of a sequence flow. Finally the precedence constraint *X is activated by Y* requires that *X* is activated by *Y* by means of a message flow. Two simple examples of precedence constraint are provided by requirements (b) and (c).

Precedence constraints can be encoded in Description Logics using specific BPMNO roles which formalize the connection between graphical objects. In particular the key roles we can use are:

- has_sequence_flow_source_ref and has_sequence_flow_target_ref.
- has_message_flow_source_ref and has_message_flow_target_ref.

These roles represent the SourceRef and TargetRef attributes of BPMN and identify which graphical elements the connecting object is connected from and to respectively. The first two roles refer to connecting object which are sequence flow, while the other two roles refer to message flow.
Constraint (b) can be formalized in DL by means of two statements

$$\begin{aligned}\mathsf{BDO{:}to\_provide\_sensible\_data} \sqsubseteq \forall \mathsf{BPMN{:}has\_sequence\_flow\_target\_ref}^{-}.\\ \forall \mathsf{BPMN{:}has\_sequence\_flow\_source\_ref.BDO{:}to\_read\_policy}^{*}\end{aligned} \tag{10}$$

$$\begin{aligned}\mathsf{BDO{:}to\_read\_policy}^{*} \equiv \neg \mathsf{BPMN{:}start\_event} \sqcap ((\mathsf{BDO{:}to\_read\_policy} \sqcap\\ \mathsf{BPMN{:}activity}) \sqcup \forall \mathsf{BPMN{:}has\_sequence\_flow\_target\_ref}^{-}.\\ \forall \mathsf{BPMN{:}has\_sequence\_flow\_source\_ref.BDO{:}to\_read\_policy}^{*})\end{aligned} \tag{11}$$

The statements above use has_sequence_flow_source_ref and has_sequence_flow_target_ref, together with an auxiliary concept $\mathsf{BDO{:}to\_read\_policy}^{*}$. In a nutshell the idea is that the concept BDO:to_provide_sensible_data is immediately preceded, in all paths defined by a sequence flow, by a graphical object of type $\mathsf{BDO{:}to\_read\_policy}^{*}$. This new concept is, in turn, defined as a graphical object which is not the start event and either it is an activity of type BDO:to_read_policy or it is preceded in all paths by

BDO:to_read_policy*. By replacing BDO:to_provide_sensible_data, BDO:to_read_policy, and BDO:to_read_policy* with X, Y and Y^* in (10) and (11) we can obtain a general encoding of constraints of the form *X is always preceded by Y*. In addition by replacing $\forall$ with $\exists$ we can obtain an encoding of *X is once preceded by Y*.

Note that, if we replace the constraint (b) with a simpler constraint of the form *"the activity of providing personal data is always **immediately** preceded by an activity of reading the policy of the organization"* then, we do not need to introduce the auxiliary concept Y^* and the encoding is directly provided by the statement

$$\begin{aligned}\text{BDO:to\_provide\_sensible\_data} \sqsubseteq\ & \forall \text{BPMN:has\_sequence\_flow\_target\_ref}^{-}.\\ & \forall \text{BPMN:has\_sequence\_flow\_source\_ref}.\text{BDO:to\_read\_policy}\end{aligned} \tag{12}$$

To formalise (c) we need to check that the activities annotated with BDO:to_read_policy are activated by an intermediate event (message) which refers to a message flow which originates by an activity of type BDO:to_provide_policy_data:

$$\begin{aligned}&\text{BDO:to\_read\_policy} \sqsubseteq \exists \text{BPMN:has\_sequence\_flow\_target\_ref}^{-}.\\ &\quad \exists \text{BPMN:has\_sequence\_flow\_source\_ref}.(\\ &\qquad \text{BPMN:intermediate\_event} \sqcap \exists \text{BPMN:has\_message\_flow\_target\_ref}^{-}.\\ &\qquad \exists \text{BPMN:has\_message\_flow\_source\_ref}.(\\ &\qquad\quad \text{BDO:to\_provide\_policy\_data} \sqcap \text{BPMN:activity}))\end{aligned} \tag{13}$$

Again by replacing the specific BDO concepts with X and Y we can obtain a schematic encoding of constraints of the form *X is activated by Y*.

Combining different constraints. By combining containment, enumeration and precedence constraints, we can encode more complex requirements. An example is provided by the following requirement

All the paths that originate from the exclusive gateways contained in the On-line shop pool must start with an event which comes from the Customer pool.

Due to expressiveness limitation imposed by Description Logics and by the fact that we want to remain in a decidable version of OWL, there are also constraints on the static part of the BPMN diagram which are are not represented in our approach. In particular all the properties that, once translated into first order logic, require more than two variables. A typical example of this kind of constraint is the fact that *X mutually_excludes Y*, that is that X and Y are always preceded by the same exclusive gateway. In fact we can express this property as follows, where $<$ is used to indicate the precedence relation between graphical elements:

$$\begin{aligned}\forall X.\forall Y.\exists Z(&xor(Z) \wedge precede(Z,X) \wedge precede(Z,Y) \wedge \forall W.(precede(Z,W) \wedge\\ &precede(W,X) \wedge precede(W,Y)) \rightarrow \neg gateway(W))\end{aligned} \tag{14}$$

We can instead represent a weaker version of the constraint above, which states that X and Y are immediately preceded by an exclusive XOR gateway as in our BPD in Figure 1, by using precedence constraints. Similar limitations apply to constraints involving parallel gateways.

5 Checking Constraints as a Satisfiability Problem

Given an Abox A_β which contains the OWL representation of a semantically annotated BPD β, we can reduce the problem of checking constraints over a BPD to a satisfiability problem in DL. In particular we can reformulate the fact that A_β represents a business process correctly annotated according to Merging_Axioms(BPMNO, BDO) and which satisfies a set of process specific constraints Constraints(BPMNO,BDO) as the fact that

$$\text{BPMNO} \cup \text{BDO} \cup \text{Merging\_Axioms}(\text{BPMNO}, \text{BDO}) \cup \text{Constraints(BPMNO,BDO)} \cup A_\beta$$

is a consistent knowledge base. In this section we illustrate how we can support the automatic transformation of a semantically annotated BPD β into the corresponding Abox A_β. We also provide an experimental evaluation of the performance of the tool, and of reasoning systems used to check structural constraints over the BPD.

5.1 Automatically Encoding a BPD into an Abox

We developed a tool for the automated transformation of a BPD into an OWL Abox. Given BPMNO, BDO, Merging_Axioms(BPMNO,BDO) and an annotated BPD β, the tool creates the Abox A_β and populates the ontology with instances of BPMN elements belonging to the specific process.

The input BPMN process is currently described in a `.bpmn` file, one of the files generated by both the Eclipse SOA Tools Platform and the Intalio Process Modeler tools. The `.bpmn` file is an XML file that contains just the structural description of the process, leaving out all the graphical details. The ontology is populated by parsing the file and instantiating the corresponding classes and properties in the BPKB Tbox.

The mapping between the XML elements/attributes used in the `.bpmn` file of the Eclipse tool and concepts and properties in the BPKB Tbox is realized by means of a mapping file. It associates each XML element of the `.bpmn` file to the corresponding concept in BPMNO and each of its attributes and XML elements (with a correspondence in the BPMNO) to the corresponding concept or property. The fragment of mapping file in Figure 5 shows the correspondences between the pool process element (i.e. the XML element having type `bpmn:Pool` in the `.bpmn` file) and the BPMNO. Each XML element of this type in the `.bpmn` file will be translated into an instance of the concept BPMNO:Pool. The values of its attributes `name` and `documentation` will be the target of the two data properties of the concept BPMNO:Pool, respectively has_swimlane_name and has_BPMN_element_documentation. Moreover, the two target objects (instances of the classes BPMNO:Object and BPMNO:Process) of the BPMNO:Pool object properties has_BPMN_element_id and has_pool_process_ref, will be instatiated by exploiting the unique id of the process element `pool`. Finally, the XML elements contained in the element `pool` and annotated with the XML tags `lanes` and `vertices` will respectively be the target objects of the BPMNO:Pool object property has_pool_lanes and of the BPMNO:Process object property has_process_graphical_elements.

The BPMN process descriptions currently generated by the Eclipse tool or the Intalio tool do not exhaustively cover the features provided by the BPMN specification and, therefore, the full ontology potential. Thus the mapping file is limited to the subset of the specification actually implemented by the considered tools and is also based on

```
<bpmn:Pool class="pool">
    <attributes>
        <this prop="has_BPMN_element_id" range="object"/>
        <this.object prop="this.object->has_object_id"/>
        <this prop="has_pool_process_ref" range="process"/>
        <this.process prop="this.process->has_process_name"/>
        <documentation prop="has_BPMN_element_documentation"/>
        <name prop="has_swimlane_name"/>
    </attributes>
    <relations>
        <lanes prop="has_pool_lanes" range="lane"/>
        <vertices prop="this.process->has_process_graphical_elements"
                  range="graphical_element"/>
    </relations>
</bpmn:Pool>
```

Fig. 5. A fragment of the mapping file

Table 1. Evaluation Results

Process	Process Elements	Semantic Annotations	Abox Creation	Classification of BPKB	Classification of BPKB with Constraints
PROCESS_A	79	13	185s	5s	205s
PROCESS_B	196	30	601s	20s	143s
PROCESS_C	324	56	1377s	28s	338s
PROCESS_D	629	112	4075s	750s	1141s

assumptions implicitly made by the tools. Similarly, the mapping file depends on the particular process representation adopted by these tools and must be adjusted whenever a different tool is used for process editing.

Finally the semantic annotations added to process elements and contained in the `.bpmn` file as XML elements of type `bpmn:TextAnnotation` are also used for populating the BDO concepts. By parsing the file, the process element associated to each XML element having type `bpmn:TextAnnotation` will be added as an instance of the BDO concept corresponding to the value of the semantic annotation.

Our tool uses the `org.w3c.dom` XML parsing library to manage the `.bpmn` input file, Protégé libraries to populate the resulting OWL Abox, and Pellet for reasoning.

5.2 An Experimental Evaluation

We performed some experiments in order to evaluate the performances of our tool in populating the BPKB Abox and the performance of DL reasoners for checking the satisfiablity of structural constraints. We considered four processes of increasing size and complexity. PROCESS_A is composed by the "Customer" pool of the process in Figure 1, PROCESS_B is the process reported in Figure 1, PROCESS_C extends PROCESS_B by describing the entire on-line shopping process, including the checkout part, and finally PROCESS_D is composed by doubling PROCESS_C. The number of BPMN graphical elements and of semantic annotations contained in the four BPDs is contained in Table 1. The processor used for the experiments is a 1.86GHz Intel Core 2, with 2 Gb of RAM and running Windows XP. The reasoner was Pellet 1.5.

Table 1 reports, for all the processes, the time spent by our tool to generate the corresponding Abox. It also reports the time spent to classify the $\mathcal{ALCHOIN}(\mathcal{D})$ BPKBs

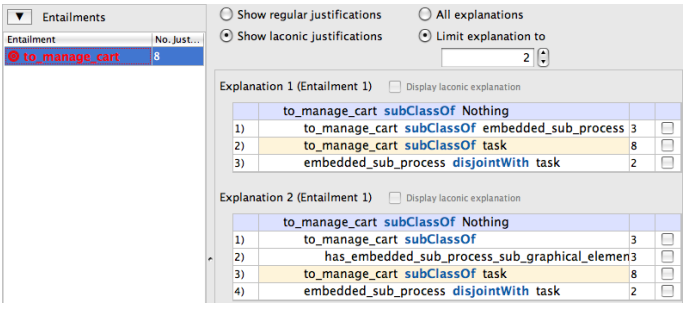

Fig. 6. Explanation generation

which encodes the annotated processes without constraints and the $\mathcal{ALCHOIQ}(\mathcal{D})$ BPKBs enriched with a set of constraints similar to the ones described in the paper by the Pellet reasoner. By looking at the results the time seems compatible with an off-line usage of the tools on medium size processes. Note that, in the case of unsatisfiable classes, which originate from conflicting requirements, an off-line usage of DL reasoners and explanation techniques similar to the ones described in [12] can be also useful to provide justifications to the business experts. Figure 6 shows an explanation obtained by using the Explanation Workbench plugin for Protege-4 of the unsatisfiable concept to_manage_cart in the case the assertion BDO:to_manage_cart $\sqsubseteq$BPMN:task is added to the knowledge base, together with constraint (4).

6 Related Work

We can roughly divide the existing proposals for adding semantics to business processes into two groups: (1) those adding semantics to specify the dynamic behavior exhibited by a business process [19,20,13], and (2) those adding semantics to specify the meaning of the entities of a BPD in order to improve the automation of business process management [2,5,18,15]. We clearly belong to the second group.

Thomas and Fellmann [18] consider the problem of augmenting EPC process models with semantic annotations. They propose a framework which joins process model and ontology by means of properties (such as the "semantic type" of a process element). Markovic [14] considers the problem of querying and reasoning on business process models. He presents a framework for describing business processes which integrates functional, behavioral, organizational and informational perspectives: the elements of the process are represented as instances of an ontology describing the process behavior (based on π-calculus), and the annotations of these elements with respect to the ontologies formalizing the aforementioned perspectives are described as relation instances. Born *et al.* [3] propose to link the elements of a business process to the elements of an ontology describing objects, states, transitions, and actions. These proposals differ substantially from ours, which establishes a set of subsumption (aka subclass or is-a) relations between the classes of the two ontologies being integrated (BPMN metamodel and domain ontology), instead of associating annotation properties to the process

instances. This difference has a direct impact on the kind of constraints that can be automatically enforced (e.g. BPMN elements annotatable by domain concepts).

De Nicola *et al.* [15] propose an abstract language (BPAL) that bridges the gap between high-level process description (e.g. in BPMN) and executable specification (e.g. in BPEL). The formal semantics offered by BPAL refers to notions such as activity, decision, etc., while the problem of integrating process model and domain ontology is not their focus. In Weber *et al.* [19], semantic annotations are introduced for validation purposes, i.e. to verify constraints about the process execution semantic. In their work, semantic annotations with respect to a background ontology are used to ensure that an executable process model behaves as expected in terms of preconditions to be fulfilled for execution and its effects. In the SUPER project [5], the SUPER ontology is used for the creation of semantic annotations of both BPMN and EPC process models in order to support automated composition, mediation and execution. Our work represents an extension of the existing literature in that we provide an ontology based approach that supports automated verification of semantic constraints defining the correctness of semantic process annotations and rich structural requirements.

7 Conclusions

In this paper we have presented an ontology-based framework to verify sets of structural constraints that involve both knowledge about the domain and the process structure. We have also described a tool for the automated transformation of an annotated business process into an OWL ontology and evaluated how standard DL reasoners can be used to automatically verify these constraints as ontology consistency violations.

Although some effort is required to create all the components of the BPKB, we envisage situations in which the business designer is mainly concerned with the specification of constraints and the validation of this specification. In particular, we foresee a situation in which the domain ontology (or an adaptation of an existing one) is available to the business designer, pre-defined high level merging axioms can be directly plugged in the system, and (candidate) annotations can be provided automatically by means of matching algorithms.

In our future work, we will work towards this objective by extending the library of "domain independent" merging axioms by examining different upper-level ontologies, including business domain ontologies. We will also investigate how to simplify the tasks of constraint specification and checking for business experts by means of more user friendly notations and tools. Finally, we will validate the approach further, on larger case studies.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes. In: VLDB 2006, pp. 343–354 (2006)

3. Born, M., Dörr, F., Weber, I.: User-friendly semantic annotation in business process modeling. In: Weske, M., Hacid, M.-S., Godart, C. (eds.) WISE Workshops 2007. LNCS, vol. 4832, pp. 260–271. Springer, Heidelberg (2007)
4. Business Process Management Initiative (BPMI). Business process modeling notation: Specification (2006), http://www.bpmn.org
5. Dimitrov, M., Simov, A., Stein, S., Konstantinov, M.: A bpmo based semantic business process modelling environment. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management at the ESWC. CEUR-WS, vol. 251 (2007)
6. Wetzstein, B., et al.: Semantic business process management: A lifecycle based requirements analysis. In: Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management. CEUR Workshop Proceedings, vol. 251 (2007)
7. 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)
8. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening Ontologies with DOLCE. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 166–181. Springer, Heidelberg (2002)
9. Ghidini, C., Hasan, M.K., Rospocher, M., Serafini, L.: A proposal of merging axioms between bpmn and dolce ontologies. Technical report, FBK-irst (2009), https://dkm.fbk.eu/index.php/BPMN_Related_Resources
10. Ghidini, C., Rospocher, M., Serafini, L.: A formalisation of BPMN in description logics. Technical Report TR 2008-06-004, FBK-irst (2008), https://dkm.fbk.eu/index.php/BPMN_Related_Resources
11. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel., D.: Semantic business process management: A vision towards using semantic web services for business process managemen. In: ICEBE 2005: Proceedings of the IEEE International Conference on e-Business Engineering, pp. 535–540. IEEE Computer Society, Los Alamitos (2005)
12. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in owl. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
13. Koschmider, A., Oberweis, A.: Ontology based business process description. In: Proceedings of the CAiSE 2005 Workshops. LNCS, pp. 321–333. Springer, Heidelberg (2005)
14. Markovic, I.: Advanced querying and reasoning on business process models. In: Abramowicz, W., Fensel, D. (eds.) BIS. LNBIP, vol. 7, pp. 189–200. Springer, Heidelberg (2008)
15. De Nicola, A., Lezoche, M., Missikoff, M.: An ontological approach to business process modeling. In: Proceedings of the 3rd Indian International Conference on Artificial Intelligence (IICAI), December 2007, pp. 1794–1813 (2007)
16. Thomas, M.F.O.: Semantic epc: Enhancing process modeling using ontology languages. In: Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management at the ESWC. CEUR-WS, vol. 251 (2007)
17. OMG. Business process modeling notation, v1.1., www.omg.org/spec/BPMN/1.1/PDF
18. Thomas, O., Fellmann, M.: Semantic epc: Enhancing process modeling using ontology languages. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM), June 2007, pp. 64–75 (2007)
19. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM) (June 2008)
20. Wong, P.Y.H., Gibbons, J.: A relative timed semantics for bpmn. In: Proceedings of 7th International Workshop on the Foundations of Coordination Languages and Software Architectures, FOCLASA 2008 (2008)