Reasoning on Semantically Annotated Processes

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

FBK-irst, Via Sommarive 18 Povo, I-38100,Trento, Italy
{dfmchiara,ghidini,rospocher,serafini,tonella}@fbk.eu

Abstract. Enriching business process models with semantic tags taken from an ontology has become a crucial necessity in service provisioning, integration and composition. In this paper we propose to represent semantically labelled business processes as part of a knowledge base that formalises: business process structure, business domains, and a set of criteria describing correct semantic labelling. Our approach allows (1) to impose domain dependent constraints during the phase of process design, and (2) to automatically verify, via logical reasoning, if business processes fulfill a set of given constraints, and to formulate queries that involve both knowledge about the domain and the process structure. Feasibility and usefulness of our approach will be shown by means of two use cases. The first one on domain specific constraints, and the second one on mining and evolution of crosscutting concerns.

1 Introduction

Semantic Business Process Management (SBPM) [16,12] 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. Focussing on process modeling, i.e. the activity of specification of business processes at an abstract level (descriptive and non executable), it has been argued that annotating process descriptions with a set of tags taken from a set of domain ontologies would provide an additional support to the business analysis in this phase. (see e.g. [20]).

However, semantic annotations will positively affect the creation of high quality process models only if they are correct. Though the notion of *correct semantic annotation* deserves a precise definition, we can intuitively say that a necessary condition for a correct annotation is that it respects types. E.g. activities in a business process should be labeled with some concepts denoting indeed an activity; similarly, conditional tests should be labeled with boolean conditions, and so on. Thus, for instance, an activity labeled "purchase order" or a gateway condition labeled "send a request" are intuitively not correct annotations. Additional requirements for a correct labeling could be imposed because of the specific application domain. For instance, in a domain in which simple actions only come from a fixed set, tasks should be only tagged with actions taken from

A. Bouguettaya, I. Krueger, and T. Margaria (Eds.): ICSOC 2008, LNCS 5364, pp. 132–146, 2008.

this set. Providing a way to specify the criteria for correct annotation is a critical and important issue in the development of business processes. It is even more important because, as shown in the examples above, while certain criteria for correct annotation can be valid for all (or a wide range of) business domains, others need to be specified for the specific business domain at hand.

Assuming that the process is correctly annotated, semantic tags can be helpful in several BPM activities. Semantic tags convey the necessary 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. In addition the search for instances of these critical patterns can be automatised using queries. Another use is related to the presence and evolution of the so-called *crosscutting concerns* [23] of a business process. Crosscutting concerns are process features that cannot be modularized into a single unit (e.g., an activity or a subprocess), because they are intrinsically scattered across the process and tangled with the other concerns. For example, in a business process for an online shop there are usually several places in which the user makes choices based on her/his preferences. All related activities form a crosscutting concern, the *user preferences* concern. Knowledge about its presence is important to understand how such a feature is currently managed and how it can be evolved in the future (e.g., storing preferences and making suggestions to the user in a proactive way). Semantic annotations provide the basic information necessary for documentation and consistent management of crosscutting concerns in business processes.

In this paper we propose an approach for (1) the specification of constraints for correct annotations of business processes, (2) the automatic verification of the correctness of annotated processes, and (3) the provision of reasoning services on labelled processes via query answering. Our approach is based on two pillars: first, on the implementation of a Knowledge Base, called *Business Process Knowledge Base*, that contains: an ontology formalizing one of the most widely used language for describing business processes, i.e. BPMN [7], a (set of) domain ontology(es) that provide the tagging language and semantics, and a set of merging axioms that specify labeling restrictions and annotation criteria; second, on a tool that automatically translates a BPMN process labelled with semantic tags into a set of assertions of the knowledge base. Detection of correctness of semantic tagging, suggestion of possible correct tagging, and semantic query based analysis of a process in order to find criticalities and manage crosscutting concerns are implemented via logical reasoning.

2 The Approach

Business Process Modeling Notation (BPMN) [21] is a (graphical) language for the specification of Business Process Diagrams (BPD). Roughly speaking, a BPD is an annotated graph whose nodes represents activities, control flows, data, and auxiliary information about the process. In our semantic variant of BPMN we allow for tagging objects of a BPD with concept descriptions taken from a (set of) domain ontology(es), i.e. shared formalizations of a specific domain.

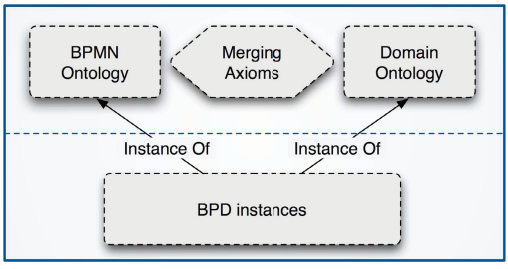

Fig. 1. Business Processes Knowledge Base

However, not every BPD object can be labelled with any concept. Meaningful semantic annotations should respect some restrictions. For instance, if the domain ontology is RosettaNet[1], then a BPMN object of type activity can be correctly tagged by rosetta:PIP[2] (Partner Interface Process, i.e. a protocol of outgoing and incoming messages representing request and response dialogue), or some of its subclasses, but it would be a mistake to label it with rosetta:TotalPrice (i.e. Total Price for the entire business document, including freight, tax and special handling if applicable).

Criteria for correct/incorrect annotation are statements that bridge the semantics of BPMN and the semantics of the domain ontology. From the formal point of view, these criteria can be represented by inclusion axioms between the concepts of an ontology formalizing BPD and the domain ontology.

In order to express this kind of constraints and to support automated reasoning on them, we propose to encode all the information about semantically annotated processes into a logical knowledge base, called *Business Processes Knowledge Base* (BPKB) and schematized in Figure 1. A BPKB is composed of the following four modules:

BPMN ontology formalises the structure of a BPD. This ontology is a formalization of the BPMN standard [21] 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.

Domain Ontology is a (set of) ontology(es) that describes a specific business domain. It can be an already existing business domain ontology (e.g. RosettaNet or similar standard business ontologies) or an ontology developed on purpose.

Merging axioms state the correspondence between the domain ontology and the BPMN ontology. They formalise the criteria for correct/incorrect semantic annotations.

BPD instances contain the description of a set of BPDs in terms of instances of the BPMN/domain ontology. Every element of the process is represented as an

[1] www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta.owl

[2] The notation <ontology>:<concept-name> stands for the concept <concept name> of the ontology <ontology>.

individual of a class. The structure of the process (i.e. the connection between different elements) is represented by means of relations between instances.

A BPKB can be implemented in any knowledge representation language with minimum expressive power and a complete decision procedure. Using logical reasoning over BPKB we can implement the following services:

Query answering on the BPD instances: a number of queries that involve either the domain ontology, the BPMN ontology or both, can be formulated. An example of query would be to "provide all the actions that are associated with credit card data". We can see that to formulate this query we need knowledge coming from the BPMN ontology (the concept "action" and the relation "associated with") and knowledge coming from the domain ontology ("credit card data").

Verification of semantic labeling: verifying whether the semantic labelling satisfies the constraints specified using the merging axioms.

Suggestions for correct labelling of the process: merging axioms can be used to suggest (sets of) labels on-the-fly during process annotation. Note that the usage of merging axioms ensures that the suggested labels are correct.

3 The Business Process Knowledge Base

We implemented BPKB using the standard semantic web language OWL (Web Ontology Language) based on Description Logics (DL). Description Logics (see [4]) are a family of knowledge representation formalisms which can be used to represent the terminological and assertional knowledge of an application domain in a structured and formally well-understood way. The terminological knowledge, contained in the so-called T-box, represents the background knowledge and the knowledge about the terminology (classes and properties) relevant for the described domain. The assertional part, the so-called A-box, contains knowledge about the individuals which populate the given domain in the form of membership statements. Roughly speaking, in our framework the terminological part - which is the stable description of a given domain - is provided by the upper level modules of Figure 1. Instead, the changeable part, which corresponds to a specific process description shown in the bottom part of Figure 1, is provided in the form of assertional knowledge.

3.1 An Ontology for BPMN

The BPMN specification [21] aims at the definition of: the building blocks of BPDs, their graphical representation, their attributes and properties, and how they can be combined to build a BPD.

Examples of BPMN elements are: *Event*, *Activity Gateway* and *Sequence Flow*. Properties of basic elements concern both the usage of the BPMN elements to compose the business process diagrams, and the behavior of the elements during the execution of a process. An example of property of the first kind is the one used to specify a *Start Event*:

"A Start Event MUST NOT be a target for Sequence Flow; it MUST NOT have incoming Sequence Flow."[An exception follows.] (1)

A different example of property, which specifies the behavioral nature of a graphical element, is the following one. This property contributes towards the specification of the exclusive nature of the Sequence Flows which originate from an Exclusive Data-Based Gateway, that is, a Gateway which is used to indicate the place where a Sequence Flow can take two or more alternative paths:

"if there are multiple outgoing Sequence Flow then only one Gate (or the DefaultGate) SHALL be selected during performance of the Process." (2)

BPMNO[3] (namely our BPMN ontology) 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 behaviour 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 modelled using formal languages for Workflow or Business Process Specification based on Petri Nets, as proposed in [18].

BPMNO is based on the latest stable BPMN specifications from OMG [21]. The ontology is structured according to the description of the complete set of BPMN Element Attributes and Types contained in Annex B of [21]. The ontology currently consists of 95 Classes and 439 Class Axioms, 108 Object Properties and 18 Object Property Axioms, and 70 Data Properties; it has the expressiveness of $\mathcal{ALCHOIN}(\mathcal{D})$ and a textual description of its Description Logic version is contained in [14].

In BPMNO, besides organizing the BPMN objects in an *is-a* taxonomy, we encoded the attributes and properties which describe how to use these elements to compose business process diagrams. As a consequence of our effort towards the modelling of properties, BPMNO contains, in its current state, more than 400 class axioms which describe a wide set of properties of the BPMN elements. Due to expressiveness limitation imposed by Description Logics and by the fact that we want to remain in a decidable version of OWL, there is a limited (and documented) number of properties listed in [21] which are not represented in BPMNO. These properties concern: (i) attributes' default values, and (ii) all the properties that, once translated into first order logic, require more than two variables. A typical example of this kind of properties is *"two objects cannot have the same object identifier"* or that *"all outgoing sequence flows connected to an inclusive gateway must have the same conditional expression attached"*.

3.2 Representing Criteria for Correct Semantic Annotations

Let us indicate with BDO a specific domain ontology contained in BPKB. Though belonging to different ontologies, concepts from BPMNO and BDO are not totally

[3] Available for download at `http://dkm.fbk.eu/index.php/BPMN_Ontology`

unrelated. Consider for instance the ontologies shown in Figure 2. A BPMN activity, for example, could correspond to BDO actions, but not to BDO objects. Such kind of relationships can be generic or domain specific constraints that a business designer decides to impose. For instance, one would like to impose that BPMN subprocesses can not be annotated by to_add_product and to_remove_product, as these two actions in BDO are considered to be atomic. Conversely, one would like to impose that certain complex BDO actions (e.g. to_manage) should correspond to subprocesses in a BPD because they must be specified in more details.

To allow the business designer to specify this kind of positive and negative constraints between pairs of concepts, each belonging to one of the two ontologies, we introduce two relations: "*annotatable only by*" ($\xrightarrow{\mathsf{AB}}$) and "*not annotatable by*" ($\xrightarrow{\mathsf{nAB}}$) from BPMNO concepts to BDO concepts. Moreover, to allow the binding of a specific BDO concept only to a certain BPMNO element, instead of defining the $\xrightarrow{\mathsf{nAB}}$ relationship between each BPMNO element and a specific BDO concept, we introduce the symmetrical relations: "*annotates only*" ($\xrightarrow{\mathsf{A}}$) and "*cannot annotate*" ($\xrightarrow{\mathsf{nA}}$). These four relationships represent constraints on the valid labelling of BPD objects. This informal description is then translated into a more formal set of DL axioms, denoted with Merging_Axioms(BPMNO, BDO), that formalise these constraints within the logical formalism. In the following table we report the intuitive meaning, and the formalization as DL axioms, of the four annotation restrictions introduced above. We use x to denote a concept of BPMNO and y to denote a concept ofBDO.

Restriction	**Intuitive meaning**	**DL axiom**[4]
$x \xrightarrow{\mathsf{AB}} y$	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 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	$y \sqsubseteq \neg x$

Figure 2 shows examples of constraints. BPMNO:data_object $\xrightarrow{\mathsf{AB}}$ BDO:object states that a BPMN data object is indeed an object of the domain ontology. BPMNO:action $\xrightarrow{\mathsf{AB}}$ BDO:activity states that an activity can be any kind of action (either an action itself or a subclass of action). However, if an activity is a subprocess, then it cannot be labelled by to_add_product and to_remove_product, as stated by BPMNO:sub_process $\xrightarrow{\mathsf{nAB}}$ BDO:to_add_product and BPMNO:sub_process $\xrightarrow{\mathsf{nAB}}$ BDO:to_remove_product. Finally the concept to_manage can be used only for annotating subprocesses, as stated by BDO:to_manage $\xrightarrow{\mathsf{A}}$ BPMNO:sub_process.

[4] 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 modelled, one may result more intuitive than the other.

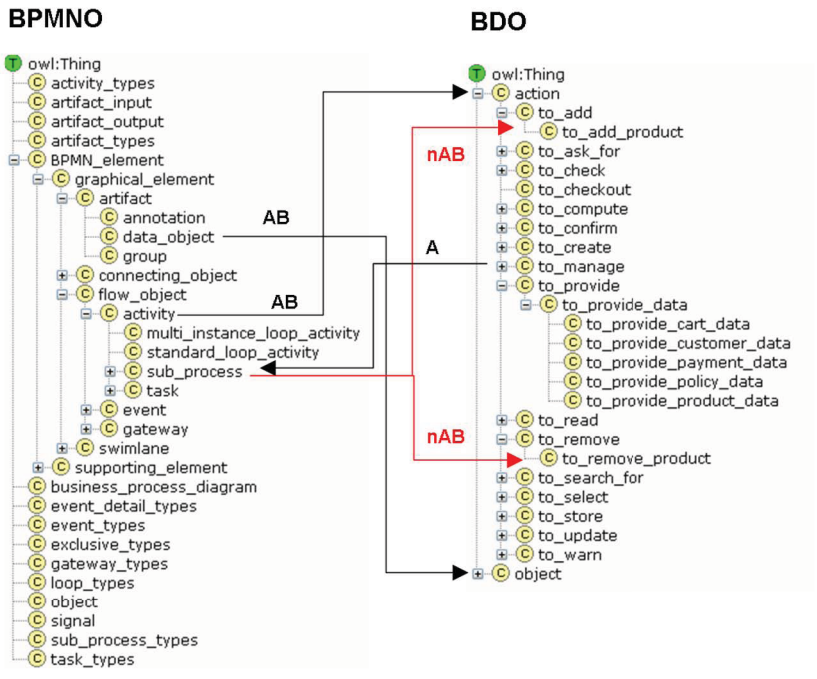

Fig. 2. Relationships between BPMNO and BDO

In the general case, some bindings specified by the user via the four primitives may generate inconsistencies in the integrated ontology. This situation can be automatically detected by verifying the consistency of $\mathsf{BPMNO} \cup \mathsf{BDO} \cup \mathsf{Merging\_Axioms}(\mathsf{BPMNO}, \mathsf{BDO})$ via a DL reasoner. In these cases, suggestions can be given automatically for recovering from inconsistency.

3.3 Representing a Semantically Annotated BPD in an OWL A-Box

Given a semantically annotated BDO β, in this section we describe how it is possible to formalize it as an A-box A_β in the language of $\mathsf{BPMNO} \cup \mathsf{BDO}$. We explain it with the help of the sample process of Figure 3. This figure represents the sub-process that manages the addition/removal of items in a shopping cart of the on-line shopping process represented in full in [13] and not included here for lack of space. Semantic annotations are preceded by the "@" symbol. The main part of the A-box associated with this sub-process is shown in Figure 4.

The elements of the A-box A_β obtained from the annotated BPD β, the so-called BPD objects in Figure 4, are all the graphical objects of β. The assertions on these elements can be divided into three groups: *BPM-type assertions*, *BPM-structural assertions* and *BPM-semantic assertions*.

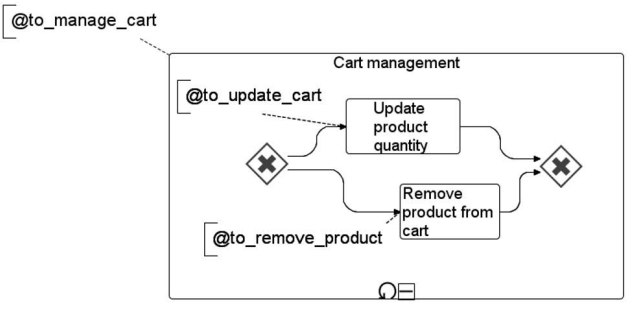

Fig. 3. A sub-process of the on-line purchase process

BPM-type assertions: for every graphical element g of type T occurring in β, A_β contains the assertions $T(g)$, i.e., g is an instance of concept T[5]. For instance the assertion sequence_flow(s_4) in Figure 4 states that the BPM object s_4 is of type sequence_flow.

BPM-structural assertions For every connecting object c of β that goes from a to b, A_β will contain two structural assertions of the form SourceRef(c, a) and TargetRef(c, b). For instance the assertion has_sequence_flow_source_ref (s_1, g_1) in Figure 4 states that the sequence flow s_1 originates from gateway g_1.

BPM-semantic assertions For every graphical element g of the diagram which is annotated with a label C (where C is a complex concept expression of the domain ontology), A_β contains the assertion $C(g)$. For instance the assertion to_update_cart(t_1) in Figure 4 states that task t_1 is an instance of concept to_update_cart and is obtained from the semantic annotation to_update_cart of the BPD in Figure 3.

Given an OWL representation A_β we can reduce the problem of checking the correctness of the semantic annotation of the BPD β to a satisfiability problem in DL. In particular we can reformulate the fact that A_β represents a business process labelled correctly as the fact that BPMNO $\cup$ BDO $\cup$ Merging_Axioms(BPMNO, BDO) $\cup A_\beta$ is a consistent knowledge base.

The semantic annotation of the process in Figure 3 is consistent with the merging axioms Merging_Axioms(BPMNO,BDO) shown in the previous section. Assume, instead, to introduce a faulty annotation by replacing @to_manage_cart in Figure 3 with @to_remove_product. In this case the assertion to_manage_cart(p_1) in Figure 4 is replaced by the assertion to_remove_product(p_1). This new assertion generates an inconsistent A-box. Indeed, BPMNO contains the axiom:

$$\text{embedded\_loop\_sub\_process} \sqsubseteq \text{sub\_process}$$

which allows to infer sub_process(p_1) from the assertion embedded_loop_sub_process (p_1). Furthermore Merging_Axioms(BPMNO,BDO) contains the annotation restriction.

[5] For the sake of readability, we omit the BPMNO prefix in non ambiguous expressions.

BPD objects	
p_1 corresponds to the entire subprocess	
$s_1, \ldots, s_4$ correspond to the four sequence flow	
g_1 and g_2 correspond to the left and the right gateways	
t_1 and t_2 correspond to the top and bottom atomic task	
BPM-type assertions	
embedded_loop_sub_process(p_1)	/* p_1 *is an iterative subprocess* */
data_based_exclusive_gateway(g_1)	/* g_1 *is a data base xor gateway* */
data_based_exclusive_gateway(g_2)	
sequence_flow(s_1)	/* s_1 *is a sequence flow object* */
sequence_flow(s_2)	
sequence_flow(s_3)	
sequence_flow(s_4)	
task(t_1)	/* s_1 *is an atomic task object* */
task(t_2)	
BPM-structural assertions	
has_embedded_sub_process_sub_graphical_elements(p_1, g_1)	
⋮	
⋮	/* p_1 *contains* g_1, g_2, $s_1 \ldots s_4$, t_1 *and* t_2 */
has_embedded_sub_process_sub_graphical_elements(p_1, t_2)	
has_sequence_flow_source_ref(s_1, g_1)	
has_sequence_flow_target_ref(s_1, t_1)	
has_sequence_flow_source_ref(s_2, g_1)	
has_sequence_flow_target_ref(s_2, t_2)	
has_sequence_flow_source_ref(s_3, t_1)	
has_sequence_flow_target_ref(s_3, g_2)	
has_sequence_flow_source_ref(s_4, t_2)	
has_sequence_flow_target_ref(s_4, g_2)	
BPM-semantic assertions	
to_manage_cart(p_1)	/* p_i *is an activity of managing of carts* */
to_update_cart(t_1)	
to_remove_product(t_2)	

Fig. 4. The encoding of the OnlineShop process in an OWL A-box

$$\mathsf{BPMNO{:}sub\_process} \sqsubseteq \neg \mathsf{BDO{:}to\_remove\_product}$$

corresponding to $\mathsf{BPMNO{:}sub\_process} \xrightarrow{\mathsf{nAB}} \mathsf{BDO{:}to\_remove\_product}$, which implies $\neg\mathsf{BDO{:}to\_remove\_product}(p_1)$. This last assertion is in contradiction with the assertion $\mathsf{BDO{:}to\_remove\_product}(p_1)$ generated by the incorrect labelling.

3.4 Automatically Encoding a BPD into an A-box

We developed a tool for the automated transformation of a BPD into an OWL A-box. Given BPMNO, BDO, Merging_Axioms(BPMNO,BDO) and a BPD β annotated with concepts taken from the domain ontology, the tool creates the A-box 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 population is realized by parsing the file and, taking advantage of a mapping file, instantiating the corresponding classes and properties in the BPKB T-Box. The mapping file associates XML tags/attributes used in the `.bpmn` file with BPMN

ontology concepts and properties. It is dependent on the particular process representation adopted by the tools generating the `.bpmn` file, hence it must be redefined or adjusted whenever a tool different from Eclipse SOA Tools Platform and Intalio Process Modeler are used for process creation and editing.

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. The mapping file is limited to the subset of the specification actually implemented by the considered tools. Sometimes the mapping is based on assumptions implicitly made by the tools (e.g. whenever a subprocess is created using the two considered tools, the concept to be instantiated is "embedded-subprocess", not "subprocess", since only embedded subprocesses can be created using the two tools).

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

4 Use Cases

Some examples of the usefulness of semantic annotations associated with a Business Process Knowledge Base are shown in the following scenarios.

Restricting to subclasses of semantically annotated BPD. A first usage of the encoding of BPMN into an ontology is the possibility to impose additional restrictions on a BPD, or on how a BPD can be semantically labelled. These restrictions can be verified automatically via reasoning in the BPKM. We propose two main forms of constraints: (1) constraints on the BPD structure, independent from the labelling; and, (2) constraints on the structure that depend on the business domain.

In the first case, a new constraint on the BPD structure can be imposed by extending BPMNO with a new set of axioms that encode this restriction. For instance, suppose that a business designer, in order to facilitate the decisions that participants in the process have to make, wants to allow only binary exclusive decisions, hence rejecting both multiple and inclusive ones. This kind of constraint can be formalized by extending BPMNO with the following DL axioms:

$$\mathsf{BPMNO{:}inclusive\_gateway} \sqsubseteq \bot \quad /^{*}\ \textit{no inclusive gateways}\ ^{*}/$$
$$\mathsf{BPMNO{:}gateway} \sqsubseteq (\leq 2)\mathsf{BPMNO{:}has\_gateway\_gate} \quad /^{*}\ \textit{gateway has at most 2 outgoing gates}\ ^{*}/$$

In the second case (i.e. restrictions that involve the business domain), the new constraints deal with business domain specific rules. For instance, in an online shop process, the business expert may be interested, for security reasons, in guaranteeing that processing payment data is preceded by a customer checkout request. Such a constraint can be formalized in DL as follows:

$$\mathsf{BDO{:}to\_provide\_payment\_data} \sqsubseteq \exists \mathsf{BPMNO{:}connect}^{-}.\mathsf{BDO{:}to\_ask\_for\_checkout}$$

where BPMNO:connect is the transitive closure of the connections provided by the connecting elements.[6] In other words, the transitive BPMNO:connect relationship ensures that there exists at least a path in the model connecting two flow objects.

Searching for Crosscutting Concerns. Another possible application of semantic reasoning over process ontologies is related to the documentation, management, and evolution of the so-called *crosscutting concerns* (CC) in business processes [25]. In a business process, a crosscutting concern is any relevant feature that cannot be modularized in a single unit (i.e. activity, subprocess, etc.) of the process description. CCs are intrinsically scattered across the process and tangled with other concerns. For example, multiple process elements usually deal with user choices and there is no unique place where all such choices are made. Hence, comprehension and modification of user choices management requires a complex, time consuming understanding of the overall process and its parts. Ontologies can help us simplify this task.

Let us consider again the On-line Shop process from which we extracted the sub-process depicted in Figure 3. A critical design decision is how to manage the customer's preferences, for example in order to store and make them available in next accesses. As observed above, this is a quite typical example of crosscutting concern in a business process, since the customer usually expresses her preferences at different points in the workflow. We can mine for such points by formulating a query matching the "customer choice" concern. Figure 5 (top left) represents such a query in SPARQL [22]. The query matches all the places where the "customer choice" concern is dealt with in the BP, collecting the five tasks that represent a customer's choice (related to products, group products, quantity and shipping method).

In a second scenario, the business designer could explicitly look for the communication between the two pools in the On-line Shop process, so as to know all the on-line shop activities that follow an event generated by a customer activity (e.g. in order to apply a new event handler). Since the process as a whole is based on the messages exchanged between the two participants, this concern is scattered across the entire process (i.e. it is a CC). The query matching the "customer-shop communication" concern is formulated as the SPARQL query shown at the bottom of Figure 5. This query asks for all the on-line shop activities triggered by (i.e. immediately following) a customer event. Similarly to the "annotatable only by" constraints considered above, we can think to provide aid for an easier query formulation (e.g. by means of a visual query language [25]) and result representation (e.g. by highlighting the query results across the process), so as to avoid exposing SPARQL directly to the final user. Once a query capturing a CC of interest has been carefully formulated, it can be recorded as a form of process documentation. Whenever understanding such a CC will be important for some process design or evolution task, re-execution of the stored query will

[6] The relation BPMNO:connect can be formalized by the following axioms: BPMNO:has_sequence_flow_source_ref^{-} $\sqsubseteq$ BPMNO:connect, BPMNO:has_sequence_flow_target_ref $\sqsubseteq$ BPMNO:connect, and Trans(connect).

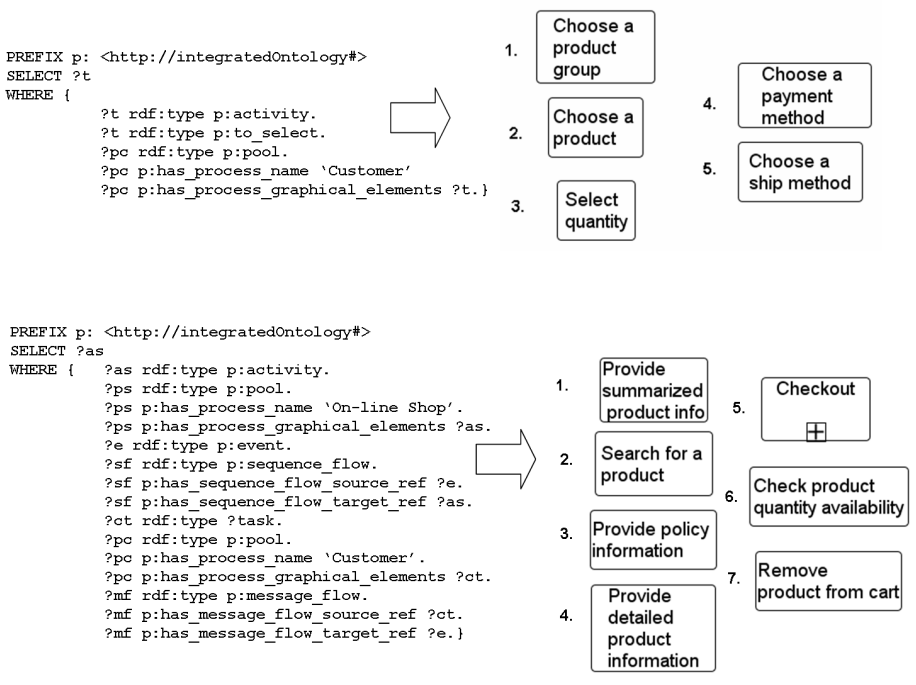

Fig. 5. Queries (on the left) and results (on the right)

immediately provide all involved process elements, even though they are possibly scattered and tangled with other process features.

5 Related Work

The idea of adding formal semantics to business processes is not new and a few approaches are already available for BPMN [26,10,27,18,5,11,24,19]. We can roughly divide the existing proposals into two groups: (1) those adding semantics to specify the dynamic behavior exhibited by a business process [26,27,18], 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 [5,11,24,19]. We clearly belong to the second group.

Thomas and Fellmann [24] 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). This differs substantially from our proposal, which establishes a set of subsumption (aka subclass or is-a) relations between the classes of the two ontologies being integrated (BPMN meta-model 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). In particular, our approach supports automatic checking of the correctness of the semantic annotation. De Nicola *et al.* [19] 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 the SUPER project [11], 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. In [26], semantic annotations are introduced for validation purposes, i.e. to verify constraints about the process execution semantics. Our work represents an extension of the existing literature in that we provide an ontology integration scheme, based on hierarchical ontology merge, that supports automated verification of semantic constraints defining the correctness of semantic process annotations. Moreover, our proposal is compatible with top-level ontologies and allows rich and expressive queries, such as those necessary to identify, document and manage crosscutting concerns in business processes.

Other related works are in the area of crosscutting concerns in business processes, which has been mainly investigated with reference to executable process description languages. AO4BPEL [3] is a dynamic aspect oriented extension of BPEL [9], designed to be as close as possible to the aspect-oriented programming language AspectJ. Similar approaches, mainly differing from AO4BPEL for the choice of the advice language (Java), have been proposed by Courbis and Finkelstein [8] and by Verheecke, Cibràn and Jonckers [2], who defined WSML (Web Service Management Layer), based on a dynamic aspect oriented extension of JAsCo [1]). Padus [6] by Braem et al. is another aspect-oriented BPEL extension, without dynamic weaving and with some improvements that increase portability.

All these works mainly focus on the developers' perspective, without considering the business designers' one. Only a few attempts are available which try to address the problem of crosscutting concerns in business processes at an abstraction level similar to the business designers' one [17,15]. Though applied to a formal modeling language like Petri Nets, the approach proposed by Hornung et al. [17] supports the designer in the process modeling phase by suggesting a list of correct and fitting process fragments for completing the business process. It is based on business rules describing constraints on the specific business domain, similar to aspects, but expressed in a specific "if-then" syntax and defined before process modeling. Another similar, rule-based approach was proposed to support automated verification of compliance to constraints [15].

6 Conclusions

We have proposed a method to add semantic annotations to a business process, based on a set of merging axioms that connect BPMN ontology and domain ontology. Semantic annotations allow formal, automated reasoning on the elements and properties of a business process. Structural and domain specific constraints

can be expressed as axioms and can be verified as ontology consistency violations. Queries on the instances (i.e. actual process elements) can be defined to match relevant process features, such as crosscutting concerns.

In our future work, we will simplify the task of ontology merging for the final user by means of tools and algorithms that handle the typical inconsistencies generated in this step. We will also investigate user friendly notations for constraint and query specification. Another direction deals with moving from specification to executable process description languages, such as BPEL, for which the discovered CCs may be mapped to AO4BPEL aspects. Finally, we will validate the approach further, on larger case studies.

References

1. Jasco: an aspect-oriented approach tailored for component based software development. In: AOSD, pp. 21–29 (2003)
2. Aspect-oriented programming for dynamic web service monitoring and selection. In: Zhang, L.-J. (ed.) ECOWS, LNCS. vol. 3250, pp. 15–29. Springer, Heidelberg (2004)
3. Mezini, M., Charfi, A.: Aspect-oriented web service composition with AO4BPEL. In (LJ) Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 168–182. Springer, Heidelberg (2004)
4. 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)
5. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes. In: VLDB 2006, pp. 343–354 (2006)
6. Braem, M., Verlaenen, K., Joncheere, N., Vanderperren, W., Van Der Straeten, R., Truyen, E., Joosen, W., Jonckers, V.: Isolating process-level concerns using padus. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 113–128. Springer, Heidelberg (2006)
7. Business Process Management Initiative (BPMI). Business process modeling notation: Specification (2006), `http://www.bpmn.org`
8. Courbis, C., Finkelstein, A.: Towards aspect weaving applications. In: ICSE 2005: Proc. of the 27th international conference on Software engineering, pp. 69–77. ACM, New York (2005)
9. Curbera, F., Goland, Y., Klein, Y., Leymann, F., Roller, D., Weerawarana, S.: Business process execution language for web services. Web page. Version 1.0 (July 31, 2002)
10. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and automated analysis of bpmn process models (2007), `http://eprints.qut.edu.au/archive/00005969/`
11. 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)
12. 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)

13. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on semantically annotated processes. Technical report, FBK-irst (2008), http://se.fbk.eu
14. Ghidini, C., Rospocher, M., Serafini, L.: A formalisation of BPMN in description logics. Technical Report TR 2008-06-004, FBK-irst (2008)
15. Happel, H.-J., Stojanovic, L.: Ontoprocess – a prototype for semantic business process verification using swrl rules. In: Proc. of the 3rd European Semantic Web Conference (2006)
16. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: A vision towards using semantic web services for business process management. In: ICEBE 2005: Proceedings of the IEEE International Conference on e-Business Engineering, pp. 535–540. IEEE Computer Society, Los Alamitos (2005)
17. Hornung, T., Koschmider, A., Oberweis, A.: A recommender system for business process models. In: 17th Annual Workshop on Information Technologies and Systems, Decemeber (2007)
18. Koschmider, A., Oberweis, A.: Ontology based business process description. In: Proceedings of the CAiSE 2005 Workshops. LNCS, pp. 321–333. Springer, Heidelberg (2005)
19. 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)
20. Fellmann, M., Thomas, 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)
21. OMG. Business process modeling notation, v1.1, http://www.omg.org/spec/BPMN-/1.1/PDF
22. Seaborne, A., Prud'hommeaux, E.: SPARQL query language for RDF. W3C recommendation, W3C (January 2008), http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
23. Tarr, P.L., Ossher, H., Harrison, W.H., Sutton Jr., S.M.: N degrees of separation: Multi-dimensional separation of concerns. In: Proc. of the International Conference on Software Engineering (ICSE), Los Angeles, CA, USA, pp. 107–119. ACM press, New York (1999)
24. 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)
25. Tonella, P., Di Francescomarino, C.: Business process concern documentation and evolution. Technical report, FBK-irst (2008), http://se.fbk.eu
26. 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)
27. Wong, P., Gibbons, J.: A Relative Timed Semantics for BPMN (submitted, 2008), Extended version http://web.comlab.ox.ac.uk/oucl/work/peter.wong/pub/bpmntime.pdf