An ontology for the Business Process Modelling Notation

Marco ROSPOCHER, Chiara GHIDINI and Luciano SERAFINI
Fondazione Bruno Kessler—IRST, Via Sommarive 18, Trento, I-38123, Italy

Abstract. In this paper we describe a formal ontological description of the Business Process Modelling Notation (BPMN), one of the most popular languages for business process modelling. The proposed ontology (the BPMN Ontology) provides a classification of all the elements of BPMN, together with the formal description of the attributes and conditions describing how the elements can be combined in a BPMN business process description. Using the classes and properties defined in the BPMN Ontology any BPMN diagram can be represented as an A-box (i.e., a set of instances and assertions on them) of the ontology: this allows the exploitation of ontological reasoning services such as consistency checking and query answering to investigate the compliance of a process with the BPMN Specification as well as other structural property of the process. The paper also presents the modelling process followed for the creation of the BPMN Ontology, and describes some application scenarios exploiting the BPMN Ontology.

Keywords. Ontology, OWL, Semantic Web, BPMN, Business Processes

1. Introduction

The *Business Process Modelling Notation*[1] (BPMN) [1] is a state of the art (graphical) language for the specification of business processes. BPMN is the result of an agreement between multiple modelling tools vendors to use a single notation for the benefit of end-user understanding and training. Indeed, the development of BPMN was driven by two major requirements: the language had to be acceptable and usable by business analysts in the business community, and the language had to support the generation of executable processes from the notation provided. To accommodate the former, BPMN supports users in writing business process models using a simple and intuitive graphical notation, while the latter is achieved by providing a mapping of the language to the Business Process Execution Language (BPEL). The combination of these two factors contributes to justify the popularity of BPMN among business process analysts.

In this paper we describe the BPMN Ontology, an ontological formalization of the BPMN specification: all the elements, attributes, and properties of the language have been formalized in OWL-DL (the Description Logics fragment of OWL), following the details provided in the BPMN OMG Specification [2]. In details, the BPMN Ontology accurately encodes the classification of all the elements of BPMN, together with the

[1] a.k.a. *Business Process Model and Notation*

formal representation of the attributes and conditions describing how the elements can be combined to obtain a BPMN process model compliant with the BPMN Specification.

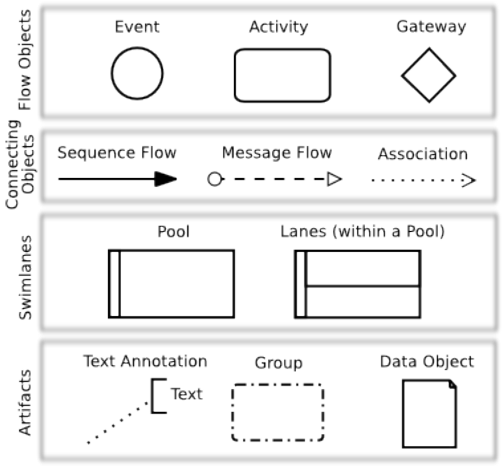
Beside providing a terminological description of the language, the classes and properties defined in the BPMN Ontology enable representing any actual BPMN process model as a set of individuals and assertions on them. This allows the exploitation of ontology-based reasoning services such as consistency checking and query answering to investigate the compliance of a process model with the BPMN Specification as well as other structural properties.

Two clarifications are needed: first, the BPMN Ontology is not intended to model the dynamic behaviour (*behavioural semantics*) of a BPMN process, i.e., how the execution flow proceeds within it. Ontology languages are not particularly suited to specify behavioural semantics. We point readers interested in behavioural semantics for BPMN to [3], [4], and [5], where proposals in terms of YAWL, algebra CSP, and UML, respectively, are presented. Instead, our contribution shares more commonalities with the work presented in [6], where a formal grammar (specifically, a context-free hypergraph grammar) for BPMN is presented. Second, the BPMN Ontology provides an ontological formalization of BPMN as a graphical language, that is, it describes all the elements of the language and how they can be used to compose BPMN diagrams. It is not intended to provide an ontological analysis of these entities in a foundational fashion (as done for instance in [7], where the suitability of BPMN as a business simulation language is investigated using the Unified Foundational Ontology [8]). To make a concrete example, the notion of activity characterised in the BPMN Ontology refers to the graphical symbol at the top of Figure 1a and is not intended to provide an ontological characterisation of the general notion of process activity that pertains to human common-sense. A preliminary investigation on possible correspondences existing between the BPMN Ontology view and a more general and foundational view, is provided by the linking effort between the BPMN Ontology and the DOLCE foundational ontology [9], briefly described at the end of Section 6 and in [10].

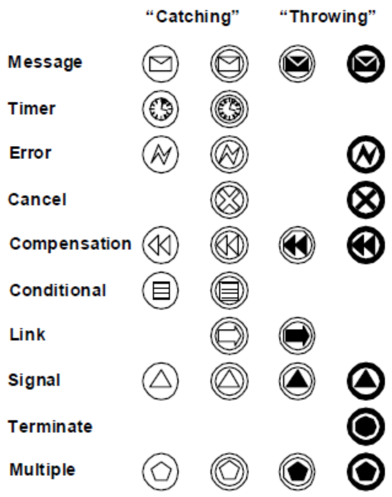
Although other BPMN ontology proposals are available (see Section 7 for more details), to the best of our knowledge, the BPMN Ontology was the first attempt to provided a high quality ontological formalization of BPMN, encoding also the constraints of the notation on how the elements can be combined to form a valid BPMN model.

Application-wise, there are many contexts where the BPMN Ontology can be exploited: to name a few, the mapping of workflows from/to different modelling notations [11], and the support to process specification in the context of semantically annotated business processes [12,10]. To give an example, the BPMN Ontology is exploited to favour knowledge interoperability in a product lifecycle management environment [13].

The paper is organized as follow. In Section 2 we provide a brief introduction to BPMN. In Section 3 we present the process that we followed for building the BPMN Ontology, and we describe in details the ontology obtained. In Section 4 we explain how any given BPMN diagram can be encoded as a set of instances, and assertions on them, in the BPMN Ontology, also presenting some key reasoning services that can exploit an instantiated BPMN Ontology. In Section 5 we describe some of the several applications that can benefit from the availability of an ontology formalizing BPMN. In Section 6 we show how our proposal address the ontology quality criteria of the Ontology Summit 2013 Communique [14]. Finally, in Section 7 we compare our work with other state of the art proposals, and present some concluding remarks.



(a)



(b)

Figure 1. The (a) BPD Core Element Set and (b) an excerpt of the BPD Extended Set (the sub-types of Event element) available in BPMN.

2. The Business Process Modelling Notation

BPMN[2] is a widely adopted graphical language for business processes modelling. A business process is described in BPMN in a Business Process Diagram (BPD), i.e., a flow-chart based representation that allows the explicit encoding of activities, control flows, data, and auxiliary information about the process.

The BPMN graphical elements that compose a BPD can be conveniently organized in two sets: the *BPD Core Element Set*, which defines the basilar elements of the language, suitable to adequately represented the basic features of most business processes, and the *BPD Extended Set*, which augments the core elements set with additional graphical elements and non-graphical attributes to allow the modelling of more complex and detailed business processes.

Figure 1a shows the content of the BPD Core Element Set, which comprises four basic categories of elements:

Flow Objects, for representing something that happens (*event*), work to be performed (*activity*), and control flow elements (*gateway*);

Connecting Objects, for showing the order in which activities are performed (*sequence flow*), the flow of messages between business entities (*message flow*), and additional information attached to flow objects (*association*);

Swimlanes, for describing participants in a process (pool), and to organize and categorize activities (lane);

Artifacts, for representing data processed/produced by activities (*data object*), informal grouping of activities within the same category (*group*), and descriptive textual notes (*text annotation*).

[2]The description here presented refers to version 1.1 of BPMN, the stable version in use at the time the ontology was developed

Attributes	**Description**
ErrorCode : String	*For an End Event:* If the Result is an Error, then the ErrorCode MUST be supplied. This "throws" the error. [..]

Table 1. Attribute ErrorCode for the the Error Event element (excerpt)

Figure 1b shows some examples of graphical elements (in particular sub-types of Event) considered in the BPD Extended Set, while Table 1 shows an excerpt of the definition of the attribute ErrorCode for the Error Event element.

The BPMN Specification [2] provides a detailed description of all the BPMN elements, their graphical representation, their attributes and properties, and how they can be combined to build a valid BPD. The BPMN specification also describes a mechanism to generate an executable business process from a BPD by means of a mapping of BPMN to a variant of the Business Process Execution Language (BPEL) [15].

3. The construction of the BPMN Ontology

The BPMN Ontology was built with some clear goals in mind: (i) to provide an OWL-DL formalization of all the elements, attributes, and properties of the language, and (ii) to enable to represent any BPMN process model as an instantiation of the ontology, such that all the structural aspects of the given process are encoded in it. Concretely, we identified some intended uses of the ontology to be developed:

checking the compliance of a process against the BPMN specification, i.e., to verify that a given process has been correctly specified according to the BPMN guidelines (e.g., the process has at least one starting event and one end event, constructs are combined in the correct way);

checking additional application-specific design guidelines. In some applications, there may be additional design guidelines for BPMN processes that need to be followed: e.g., guidelines to guarantee process readability could require that every diagram should not contain more than ten sub-processes, or that every gate should have at most three out-going flows. We would like to be able to express these guidelines as formal constraints on top of the ontology, in order to check them automatically;

semantic description and retrieval of processes (or process elements). We would like to be able to annotate any process element with a semantic description, defining the domain semantic of the element, and consequently to be able to query processes using such semantic annotations. For instance we would like to be able to state that a certain sub-process is of type "privacy critical", and we would like to be able to retrieve all processes that contains privacy critical sub-processes, or all privacy critical activities within a process.

To better shape the scope of the ontology according to these intended uses, we relied on *competency questions* [16], that is questions that an (instantiated) ontology should be able to answer, and that therefore are useful to guide the formalization process. Some of the competency questions considered are:

- "*How many flow elements does process X contain?*",
- "*What is the error code associated to error event W?*",

- "*What type of BPMN elements does sub-process Y in process X contains?*",
- "*What is the BPMN element connected by a sequence flow to activity Z?*",
- "*Is there a path of sequence flows connecting activity Z_1 to activity Z_2?*",
- "*Is process XYZ a valid process according to the BPMN specification?*".

The development of the BPMN Ontology was driven and facilitated by the availability of a well-organized and complete description of the language, the BPMN Specification. Each BPMN element is accurately described in a corresponding section of the BPMN Specification, where the following content is usually provided:

- an introductory description of the element, together with some general properties and conditions about it;
- a compact tabular description of the element attributes: in particular, for each attribute, its name, its value type, its multiplicity details, and a description of some specific conditions to be met for its instantiation are provided;
- a detailed description of the conditions holding for connecting the current element with other elements of the language;
- additional details on execution level aspects of the element, describing in particular how the process flow progresses through the element when the process is executed.

To build the BPMN Ontology we followed the three-step modelling process here described. **First** (*signature identification*), we manually processed the BPMN Specification to identify all the elements of the language: we associated each of these elements to a class in the ontology. Furthermore, guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology classes encoding the BPMN elements. For example, we defined that class FlowObject has three subclasses, namely Event, Activity and Gateway.

Second (*attribute restriction*), we considered the attributes table corresponding to each element. Each attribute defined in the table was formalized either as a datatype property or an object property, based on the following general criteria:

1. *the value type of the attribute is another BPMN element* (e.g., the attribute Target of the BPMN element Intermediate Event has as value type the BPMN element Activity [2, p. 47]): we formalized the attribute as an object property having as property domain the class associated to the current element, and as property range the class corresponding to the element mentioned as value type of the attribute.
2. *the value type of the attribute is a datatype, but only an enumerated set of options is allowed and some conditions may apply to these options* (e.g., the attribute AdHocOrdering of the BPMN element Embedded Sub-Process has as value type *String*, but only the two values *Sequential*, *Parallel* area allowed [2, p. 58]): we formalized the attribute as an object property having as property domain the class associated to the current element, and as property range a new class characterized by the enumeration all possible values of the attribute;
3. *the value type of the attribute is a datatype with no restriction* (e.g., the attribute Text of the BPMN element TextAnnotation has as value type *String* [2, p. 35]): we formalized the attribute as a datatype property having as property domain the class associated to the current element, and as property range an OWL datatype compatible to the one specified in the value type of the attribute.

For each attribute, we formalized its multiplicity details as an OWL cardinality restriction on the class having the attribute. The formalization adopted is a straightforward one (see e.g., [17]): (0..1) multiplicity is encoded as "at most one" OWL cardinality restriction, (1) multiplicity is encoded as "exactly one" OWL cardinality restriction, (1..n) multiplicity is encoded as "at least one" OWL cardinality restriction, while the (0..n) multiplicity is not encoded at all. For example, the (0..1) multiplicity of the State attribute of BPMN element DataObject [2, p. 94] is formalized in the BPMN Ontology as the OWL cardinality restriction:

$$\mathsf{DataObject} \sqsubseteq (\leq 1)\mathsf{hasState} \tag{1}$$

We also encoded additional conditions ruling the usage of the attribute. An example of conditions expressed on attributes is the following [2, p. 51]:

“If the result of an End Event is an Event Detail of type Error, than the Error Code for the Event Detail MUST be supplied”. (2)

Due to the variety of the typology of conditions described in the BPMN Specification, their formalization was performed case by case. For the example considered in (2), the formalization proposed is the following:

$$\mathsf{EndEvent} \sqsubseteq (\neg\mathsf{hasResult.Error}) \sqcup \mathsf{hasResult.}(\mathsf{Error} \sqcap \exists\mathsf{hasErrorCode}) \tag{3}$$

In the **third** (*structural constraints formalization*) and final step of the modelling process, we formalized the conditions concerning the usage of the BPMN elements to compose a BPD. An example of condition of this type is the following [2, p. 51]:

“A Start Event MUST be a source for Sequence Flow”. (4)

A further example is the following one, which applies to Gateway [2, p. 72]:

“If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates”. (5)

Again, due to the variety of these conditions, their formalization was performed case by case. For the examples considered in (4) and (5), the formalization proposed is the following:

$$\mathsf{StartEvent} \sqsubseteq \exists\mathsf{hasConnectingObjectSource}^{-1}.\mathsf{SequenceFlow} \tag{6}$$

$$\begin{aligned}\mathsf{Gateway} \sqsubseteq & (\geq 2)\mathsf{hasSequenceFlowTarget}^{-1} \sqcup \\ & ((\leq 1)\mathsf{hasSequenceFlowTarget}^{-1} \sqcap (\geq 2)\mathsf{hasGatewayGate})\end{aligned} \tag{7}$$

Although many of the properties and conditions described in the BPMN Specification have been formalized, a few documented ones are not encoded in the BPMN Ontology. We can classify them in the following three groups.

Execution level properties These properties specifies the behavioural nature of the graphical elements in a BPD. An example of a property of this type is the following one, that 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 (see [2, p. 77]):

> *"If there are multiple outgoing Sequence Flows then only one Gate (or the DefaultGate) SHALL be selected during performance of the Process".* (8)

The BPMN Ontology provides a formalization of the structural part of BPDs, i.e., which are the elements of a BPD and how they can be connected. The BPMN Ontology is not intended to model the dynamic behaviour of BPDs (that is, how the flow proceeds within a BPD), and therefore execution level properties were not considered in the formalization.

Attributes default values The value taken by the required attributes when they are not explicitly assigned in the process model. For example, the following condition is expressed for the Implementation attribute (allowed values are: Web Service, Other, and Unspecified) of Message EventDetail element [2, p. 52]:

> *"A Web service is the default technology".* (9)

Attributes default values have not been formalized because OWL does not support the specification of properties default values (we recall that attributes are formalized as properties in the BPMN Ontology).

Undecideable conditions Due to expressiveness limitation imposed by Description Logics and by the fact that we wanted to remain in a decidable fragment of OWL, there is a limited number of conditions described in the BPMN Specification which are not represented in the BPMN Ontology. A typical example is the following property concerning Inclusive Gateway elements [2, p. 73]:

> *"The ConditionExpression attribute MUST be unique for all the outgoing Sequence Flows connected to an Inclusive Gateway".* (10)

This conditions could be formalized by defining a functional property as the chain of two other properties, the one connecting the gateways to sequence flows and the one associating a condition to a sequence flow, but imposing number restrictions on property chains (or, more generally, on complex roles) lead to undecidability, as observed in [18].

The BPMN Ontology Table 2 reports some details about the version of the BPMN Ontology currently available.[3] The BPMN Ontology contains a considerable amount of class axioms (463) compared to the number of classes available (117): this is a consequence of the considerable effort towards the representation of the wide set of properties and conditions specified in the BPMN Specification, needed to encode the constraints on the structure of BPMN business processes.

[3] Version 1.01 [September 2009]. Available for download at `http://dkm.fbk.eu/rospocher/resources/bpmn/BPMNOntology.html`. A textual description in the formalism of Description Logics is provided in [19].

DL Expressivity	$\mathcal{SHOIN}(\mathcal{D})$
Classes	117
Object / Datatype Properties	123 / 48
Individuals	104

Annotations	504
Class Axioms	463
Object / Datatype Property Axioms	236 / 96
Individual Axioms	250

Table 2. BPMN Ontology metrics

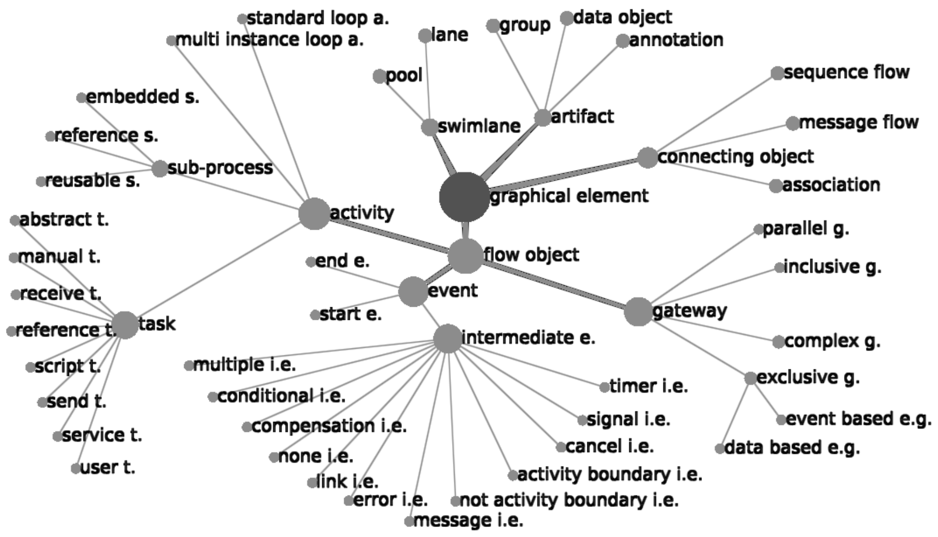


Figure 2. An overview of the taxonomy of BPMN graphical elements (root: graphical element) as defined in the BPMN Ontology (some labels are shortened to favour figure readability).

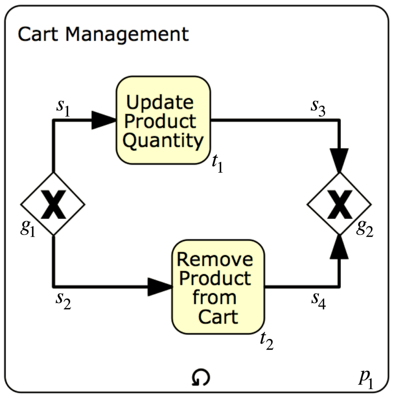
The core component of the BPMN Ontology is the set of BPMN Elements, divided into two disjoint classes graphical element and supporting element. Graphical element contains the main elements used to describe BPDs, and in particular the BPD Core Element Set, then further specified in terms of sub-classes. For instance, connecting object is the disjoint union of the classes sequence flow, message flow, and association, while flow object is the disjoint union of the classes activity, gateway, and event. A graphical representation of the hierarchy of the classes of the BPMN Ontology rooted at graphical element is shown in Figure 2.

Supporting element instead contains 16 additional types of elements, and few additional subclasses, mainly used to specify the attribute values of graphical objects. For instance, the supporting element input set is used to define an attribute value of the graphical object activity which describes the data requirements for the input of the activity.

The entities of the BPMN Ontology have been documented with selected relevant text taken from the BPMN Specification and details about particular modelling choices performed, provided as rdfs:label and rdfs:comment annotation.

4. Instantiating a BPD in the BPMN Ontology

Given a business process model described in a BPD, it is possible to represent it as an A-box in the language of the BPMN Ontology, i.e., a set of individuals, and assertions on them, instantiated according to the BPMN Ontology.



(a)

BPD individuals	
p_1 corresponds to the entire subprocess	
$s_1, \ldots, s_4$ correspond to the four sequence flows	
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	
Type assertions	
embedded_loop_sub_process(p_1)	
data_based_exclusive_gateway(g_i)	$i = 1,2$
sequence_flow(s_j)	$j = 1,..,4$
task(t_k)	$k = 1,2$
Structural assertions	
has_graphical_elements(p_1, s_i)	$j = 1,..,4$
has_graphical_elements(p_1, g_i)	$i = 1,2$
has_graphical_elements(p_1, t_i)	$k = 1,2$
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)	

(b)

Figure 3. (a) the excerpt of an on-line shopping BPD (Cart Management Sub-process) and (b) its encoding in an OWL A-box of the BPMN Ontology.

We explain in details the instantiation process with the help of the sample BPD of Figure 3a, an excerpt of an on-line shopping process model representing the sub-process that manages the addition/removal of items in a shopping cart. The main part of the BPD A-box associated with this sub-process is shown in Figure 3b. For the sake of simplicity, we limit the exposition to graphical elements, although a similar mechanism also holds for the other BPMN elements (e.g., attributes).

The elements of the BPD A-box, the so-called *BPD individuals* in Figure 3b, are all the graphical objects of the BPD. The assertions on these elements can be divided into two groups: *type assertions* and *structural assertions*.

type assertions: for every graphical element g of type T occurring in the BPD, the BPD A-box contains the assertions $T(g)$, i.e., g is an instance of concept T. For instance, the assertion sequence_flow(s_4) in Figure 3b states that the BPD object s_4 is of type sequence_flow.

structural assertions: for every connecting object c of the BPD, where c goes from object a to object b, the BPD A-box 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 3b states that the sequence flow s_1 originates from gateway g_1.[4]

The BPD A-box instantiation process can be easily automatized: a tool to automatically formalize business process models developed with the Eclipse SOA Tools Platform and

[4]See `http://selab.fbk.eu/difrancescomarino/SemanticBPM/` for some examples of instantiated BPMN Ontology.

the Intalio Process Modeler was presented in [10], while a plug-in for Protégé was developed to allow the design of simple business process models and their automatic instantiation according to the BPMN Ontology [20].

By encoding a BPD as a set of instances of the BPMN Ontology, several reasoning services over an instantiated BPMN Ontology can be implemented, among them:

Query answering on BPD instances Given an instantiated BPMN Ontology, it is possible to provide process querying mechanisms that exploits via reasoning the information formalized in the BPMN Ontology: "*Which are the activities which follows gateways and produce a data object?*" and "*Are there sub-processes which do not contain start/end events?*" are example of queries that may be easily encoded in SPARQL and run against the BPMN Ontology instantiated with the process considered.

Compliance checking of a BPD against the BPMN Specification The BPMN Ontology may also be applied to verify the compliance of a process model with the structural conditions enforced by BPMN on BPDs, like e.g., conditions (2), (4), and (5) reported in Section 3. Given a BPD, this verification can be performed by checking via OWL reasoning the consistency of the corresponding instantiated BPMN Ontology. However, the usage of OWL reasoning for this purpose requires some carefulness. This because OWL semantics is based on the *Open World Assumption (OWA)* - i.e., the knowledge of the world is incomplete, thus a fact cannot be inferred to be false on the basis of a failure to derive it - and does not satisfy the *Unique Names Assumption (UNA)* - i.e., two entities with different identifiers are distinct objects - that make complicate to use OWL for data validation where complete knowledge can be assumed (i.e., a *closed world*), like in the case of a BPD. For example, OWL allows the encoding of requirement (5), as shown in the DL axiom (7), but having a gateway with an incoming sequence flow and no gates would not cause the instantiated BPMN Ontology to be logically inconsistent, due to the Open World Assumption. However, as discussed in [21], it is possible to define an *Integrity Constraint* (IC) semantics for OWL axioms in order to enable closed world constraints validation. More details are provided in [22].

5. Application Scenarios

There are a number of contexts and purposes in which the BPMN Ontology turns out to be useful. The BPMN Ontology can be exploited in (Semantic Web) applications where a structured and terminological description of BPMN is needed, for instance for information retrieval of resources annotated with respect to the elements of the language. Analogously, the BPMN Ontology may be employed in frameworks where workflows encoded in BPMN have to be translated in a different process model formalism and vice versa: the translation may be based on ontology mappings between the BPMN Ontology and an ontology formalising the target language, in the spirit for instance of the approach presented in [11]. Indeed, the BPMN Ontology is effectively exploited to favour knowledge interoperability in a product lifecycle management environment [13].

Particularly interesting are the applications exploiting the instantiation of the BPMN Ontology. For instance, the possibility to check the compliance of a BPD against the BPMN Specification in an automatic manner can effectively support the process modelling activities. Actually, the compliance checking described in Section 4 can be further extended to included additional structural conditions (*constraints*) on BPDs to those directly enforced by the BPMN Specification, like important structural requirements that a specific process model under construction has to obey. These requirements may have various nature and rationale: to keep the model as simple and readable as possible, to ensure that exceptional situations are always handled, and so on. For example, the following structural requirements may me imposed on a given process model:

"Gateways in the process may have at most two outgoing sequence flows"; (11)

"For every error event attached to the boundary of an activity (i.e., capturing an exception), a sequence flow connecting it to an activity (i.e., an error handling activity) should be provided". (12)

These constraints can be formalized as DL inclusions axioms and, by enriching the BPMN Ontology with them, it is possible to automatically verify via ontology reasoning whether a given BPD also satisfies the given structural requirements.

A further application exploiting the instantiation of the BPMN Ontology and the reasoning services that can be offered on it concerns *semantically annotated business processes*, i.e., business process models enriched with semantic annotations taken from a domain ontology. These additional semantic information allows the improvement of the level of automation of many process specification related activities, thus providing an additional support to the business analyst in the modelling phase [23]. [12] describes a framework in which all the information about semantically annotated business processes is encoded into a logical knowledge base, called *Business Processes Knowledge Base (BPKB)*, having the BPMN Ontology as one of its main modules: the individuals in the BPKB represent an actual annotated business process model, and are instantiated with respect to both the BPMN Ontology and a domain ontology, used for the annotation, describing the domain at hand. The proposed framework allows for offering several reasoning based services: to name a few, rich query answering on the BPD instances (with queries that may involve both BPMN Ontology and the domain ontology, e.g., "*provide all the actions that are associated with credit card data*"), verification of the semantic annotations provided, and suggestion for correct process annotation. The framework has been further enriched in [10] to allow the expression and verification of process model structural constraints combining both domain specific knowledge and the structural information encoded in the BPMN Ontology, such as:

"The activity of providing personal data is always preceded by an activity of reading the policy of the organization". (13)

We conclude mentioning that since June 2009, when the BPMN Ontology was released, we monitored an average of 25 downloads per month, a quite remarkable value considering the high specificity of the ontology, thus suggesting that the BPMN Ontology may have been employed in many more application settings than those we are aware of.

6. How the BPMN Ontology address the Ontology Summit 2013 criteria

The BPMN Ontology was developed a few years before the proposal of the five evaluation criteria by the 2013 Ontology Summit [14]: nonetheless, we argue that the very same criteria drove its development.

Concerning intelligibility (*Can humans understand the ontology correctly?*), we remark that we annotated the elements (classes, object and datatype properties, individuals) of the BPMN Ontology with relevant text taken from the BPMN Specification. These labels aim to favour the "readability" of the ontology also by domain experts, as well as its exploitation in applications that may show explicitly the elements of the ontology.

As previously mentioned, the development of the BPMN Ontology was guided and facilitated by the BPMN Specification: the axioms encoded in the ontology correspond to explicit statements described in the BPMN Specification. Although typically this kind of "translation" is error-prone, due to the intrinsic ambiguity of natural language, we remark that in our case this phenomena is less prominent, due to the technical (i.e., more controlled) language used in the BPMN Specification. Furthermore, the annotations used in the ontology are actual excerpts of the BPMN Specification, and hence accurate to document the entities of the BPMN language. Moreover, in [24] we showed with automatic techniques that the BPMN Ontology provides an adequate terminological coverage of the BPMN Specification. Therefore, we believe that the ontology appropriately address the fidelity (*Does the ontology accurately represent its domain?*) criteria.

Concerning craftsmanship (*Is the ontology well-built and are design decisions followed consistently?*), we applied consistently some modelling choices, like for the formalization of BPMN element attributes (see Section 3), or the adoption of the single inheritance for subsumption (i.e., each class in the ontology has at most one parent). Furthermore, the ontology is well-documented (see also [19]).

In addition to formally represent the whole BPMN vocabulary, the BPMN Ontology covers all the elements of the BPMN language, and allows the formal representation of any BPD as a set of individuals and assertions on them, coherently with the defined competency questions. As such, it address the requirement of formally representing the structural aspect of the BPMN language (fitness criteria: *Does the representation of the domain fit the requirements for its intended use?*). Furthermore, as documented in [12,10,22], the ontology is the core element of a framework offering querying capabilities and reasoning services over semantically annotated business processes, and thus it address the deployability criteria (*Does the deployed ontology meet the requirements of the information system of which it is part?*).

Concerning the additional criteria mentioned in the FOIS 2014 Ontology Competition call for papers, we remark that, given its intended application in the context of reasoning over semantically annotated business process, we paid particular attention to the logical aspects in the development of the ontology. The logical consistency of the ontology can be easily verified with any state-of-the-art DL reasoner; at the same time, it is possible to verify the inferred class hierarchy and the additional consequences that can be produced from the ontology. Furthermore, due to the way the ontology was modelled, a BPD corresponds to an A-box of the BPMN Ontology, and therefore the ontology is satisfied by its intended models.

For what concerns the link criteria, in [25] we provided some preliminary links between the BPMN Ontology and the DOLCE foundational ontology [9], where a subsumption relation is defined between some classes of BPMN Ontology (e.g., Activity) and some classes in DOLCE (e.g., Process). As explained more in details in [10], these links (a.k.a., *merging axioms*) are meant to define criteria for the correct / incorrect semantic annotation of the elements of a BPD (e.g., *a BPMN activity can be annotated only with concepts that identifies processes according to DOLCE*). As a consequence, if a domain ontology grounded in DOLCE is used to semantically annotate a BPD, the aforementioned links automatically imply that only some classes of the chosen ontology can be used to annotate given elements of the BPD (e.g., only classes that specialize the Process DOLCE class can be used to annotate a BPMN Activity element).

7. Discussion and concluding remarks

Some other proposals of ontological formalization of BPMN are available in the state-of-the-art, thus remarking the importance of having such an artefact. The sBPMN ontology [26] was developed within the EU project SUPER. sBPMN is a WSML ontology that formalises the elements and attributes of BPMN. In comparison to sBPMN, the BPMN Ontology additionally provides a detailed formalization of the properties and conditions that state how to combine BPMN elements to form a valid BPD (e.g., conditions 2 and 5 presented in Section 3). Other proposals of ontology formalizing BPMN are mentioned in [27,28]: however, at the time of writing this paper, both ontologies cited in those works were not available for download. Furthermore, according to the description provided, they don't seem to cover the extensive set of elements, attributes, and properties, formalized in the BPMN Ontology. A further proposal of an ontology for BPMN is provided in [29]: however, the encoding is limited to the description of the taxonomy of BPMN elements. A couple of years after the release of the BPMN Ontology (2009), Natschläger [30] presented an ontological formalization of BPMN 2.0, built according to similar principles to the ones we applied for modelling our ontology.

The construction of the BPMN Ontology took about 4 person-weeks, including verification and revision. Technically speaking, the actual formalization was performed with the `latex2owl` tool,[5] a convenient way for writing ontologies with a simple textual editor. The ontology created, a text file containing axioms encoded in latex-like syntax, was then converted with the tool in a standard OWL RDF/XML serialization.

The current version of the BPMN Ontology available for download covers version 1.1 of BPMN. We are currently updating it to the latest stable version of the language, BPMN 2.02. While the update to BPMN 2.02 will modify parts of the ontology, to reflect differences in the language specification, the changes will not affect the contribution of this paper, namely, the process that we follow for building the BPMN Ontology, the encoding of BPMN diagram as a set of instances and assertions, and the services and applications that can benefit from the availability of an ontology formalizing BPMN.

References

[1] OMG. Business Process Modeling Notation. www.omg.org.

[2] OMG. Business Process Modeling Notation, v1.1 - Specification. www.omg.org/spec/BPMN/1.1/PDF.

[3] JianHong Ye, ShiXin Sun, Wen Song, and LiJie Wen. Formal Semantics of BPMN Process Models Using YAWL. In *Intelligent Information Technology Application*, volume 2, pages 70 –74, dec 2008.

[4] Peter Wong and Jeremy Gibbons. A Process Semantics for BPMN. In *Formal Methods and Software Engineering*, volume 5256 of *LNCS*, pages 355–374. Springer Berlin / Heidelberg, 2008.

[5] Oana Nicolae, Mirel Cosulschi, Adrian Giurca, and Gerd Wagner. Towards a BPMN Semantics Using UML Models. In *BPM Workshops*, volume 17 of *LNBIP*, pages 585–596. Springer, 2009.

[5]Available for download at: `http://dkm.fbk.eu/rospocher/resources/Latex2owl-v1.6.zip`.

[6] S. Mazanek and M. Hanus. Constructing a bidirectional transformation between BPMN and BPEL with a functional logic programming language. *J. of Visual Languages & Computing*, 22(1):66 – 89, 2011.

[7] Giancarlo Guizzardi and Gerd Wagner. Can BPMN Be Used for Making Simulation Models? In *Enterprise and Organizational Modeling and Simulation*. Springer, 2011.

[8] Giancarlo Guizzardi and Gerd Wagner. A unified foundational ontology and some applications of it in business modeling. In *In CAiSE Workshops (3*, pages 129–143, 2004.

[9] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening Ontologies with DOLCE. In *Proc. EKAW '02*, volume 2473 of *LNCS*, pages 166–181, 2002.

[10] C. Di Francescomarino, C. Ghidini, M. Rospocher, L. Serafini, and P. Tonella. Semantically-aided business process modeling. In *8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2009.

[11] H. Kondylakis, D. Plexousakis, H. Wache, D. Fieldkamp, K. Hinkelmann, and A. Bergmayr. Semantic Technology to enable Business and IT alignment. In *Open Knowledge Models Workshop*, 2010.

[12] Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher, Luciano Serafini, and Paolo Tonella. Reasoning on semantically annotated processes. In *Proceedings of the 6th International Conference on Service Oriented Computing (ICSOC'08)*, volume 5364 of *LNCS*, pages 132–146. Springer, 2008.

[13] Yongxin Liao, Mario Lezoche, Eduardo Loures, Hervé Panetto, and Nacer Boudjlida. Semantic Enrichment of Models to Assist Knowledge Management in a PLM Environment. In *On the Move to Meaningful Internet Systems: OTM 2013*, volume 8185 of *LNCS*, pages 267–274. Springer, 2013.

[14] F. Neuhaus, A. Vizedom, K. Baclawski, M. Bennett, M. Dean, M. Denny, M. Grüninger, A. Hashemi, T. Longstreth, L. Obrst, S. Ray, R. D. Sriram, T. Schneider, M. Vegetti, M. West, and P. Yim. Towards ontology evaluation across the life cycle. *Applied Ontology*, 8(3):179–194, 2013.

[15] F. Curbera, Y. Goland, Y. Klein, F. Leymann, D. Roller, and S. Weerawarana. Business Process Execution Language for Web Services. Web page. Version 1.0 - July 31, 2002 - Accessed January 30, 2014.

[16] M. Grüninger and M. S. Fox. Methodology for the Design and Evaluation of Ontologies. In *IJCAI '95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[17] S. Brockmans, R. Colomb, E. Kendall, E. Wallace, C. Welty, G. Xie, and P. Haase. A Model Driven Approach for Building OWL DL and OWL Full Ontologies. In *The Semantic Web - ISWC 2006*, volume 4273 of *LNCS*, pages 187–200. Springer, 2006.

[18] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, May 2000.

[19] C. Ghidini, M. Rospocher, and L. Serafini. A formalisation of BPMN in Description Logics. Technical Report FBK2008-06-004, 2008. http://dkm.fbk.eu/rospocher/resources/bpmn/BPMNOntology_DL.pdf.

[20] M. G. Spinelli. Gestione della semantica di processi BPMN attraverso un plugin per Protégé, 2009.

[21] Evren Sirin and Jiao Tao. Towards Integrity Constraints in OWL. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529. CEUR-WS.org, 2008.

[22] Chiara Ghidini, Chiara Di Francescomarino, Marco Rospocher, Paolo Tonella, and Luciano Serafini. Semantics-Based Aspect-Oriented Management of Exceptional Flows in Business Processes. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,*, 42(1):25 –37, 2012.

[23] M. Fellmann O. Thomas. Semantic EPC: Enhancing Process Modeling Using Ontology Languages. In *Proc. of the SBPM 2007 Workshop*, volume 251 of *CEUR-WS*, 2007.

[24] Marco Rospocher, Sara Tonelli, Luciano Serafini, and Emanuele Pianta. Corpus-based terminological evaluation of ontologies. *Applied Ontology*, 7(4):429–448, 2012.

[25] C. Ghidini, M. K. Hasan, M. Rospocher, and L. Serafini. A proposal of merging axioms between BPMN and DOLCE ontologies. Technical report, FBK-irst, 2009. http://dkm.fbk.eu/rospocher/resources/bpmn/MergingAxiomsBPMN-DOLCE.html.

[26] W. Abramowicz, C. Liliana, A. Filipowska, G. Pierre, N. Joerg, B. Norton, C. Pedrinaci, G. Zeissler, and S. Zoeller. D.1.5. SUPER Process Ontology Stack, Evolved Version, 2009.

[27] Linus Chow. BPMN 2.0 Primitives and Semantic Technology - Proof of Concept, 2011. Talk presented on July 13, 2011 - Slides Accessed January 30, 2014.

[28] Mohamed Keshk. Executable BPMN: BPMN-2.0 Ontology-Based Engine, 2009. Talk presented on November 3, 2009 - Slides Accessed January 30, 2014.

[29] Hui Liu. BPMN Ontology, 2011. Accessed January 30, 2014.

[30] Christine Natschläger. Towards a BPMN 2.0 Ontology. In *Business Process Model and Notation*, volume 95 of *LNBIP*, pages 1–15. Springer, 2011.