An ontology for the Business Process Modelling Notation

Marco Rospocher
Fondazione Bruno Kessler, Data and Knowledge Management Unit
Trento, Italy
rospocher@fbk.eu :: http://dkm.fbk.eu/rospocher

joint work with:
Chiara Ghidini, Luciano Serafini

8th International Conference on Formal Ontology in Information Systems (FOIS 2014)
September, 22-25, 2014 - Rio de Janeiro, Brazil
Business Process Modelling Notation (BPMN)

- State of the art graphical language for the specification of business processes
An ontology for the Business Process Modelling Notation (BPMN)

Core Element Set

- **Event** for representing something that happens (event), work to be performed (activity), and control flow elements (gateway);
- **Activity** for showing the order in which activities are performed (sequence flow), ...
- **Gateway** for describing participants in a process (pool), and to organize and categorize activities (lane);
- **Sequence Flow** for representing data processed/produced by activities (data object), informal grouping of activities (group),...
## Business Process Modelling Notation (BPMN)

### Extended Element Set (e.g. Event types)

<table>
<thead>
<tr>
<th></th>
<th>&quot;Catching&quot;</th>
<th>&quot;Throwing&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>![message_icon]</td>
<td>![message_icon]</td>
</tr>
<tr>
<td>Timer</td>
<td>![timer_icon]</td>
<td>![message_icon]</td>
</tr>
<tr>
<td>Error</td>
<td>![error_icon]</td>
<td>![error_icon]</td>
</tr>
<tr>
<td>Cancel</td>
<td>![cancel_icon]</td>
<td>![cancel_icon]</td>
</tr>
<tr>
<td>Compensation</td>
<td>![compensation_icon]</td>
<td>![compensation_icon]</td>
</tr>
<tr>
<td>Conditional</td>
<td>![conditional_icon]</td>
<td>![conditional_icon]</td>
</tr>
<tr>
<td>Link</td>
<td>![link_icon]</td>
<td>![link_icon]</td>
</tr>
<tr>
<td>Signal</td>
<td>![signal_icon]</td>
<td>![signal_icon]</td>
</tr>
<tr>
<td>Terminate</td>
<td>![terminate_icon]</td>
<td>![terminate_icon]</td>
</tr>
<tr>
<td>Multiple</td>
<td>![multiple_icon]</td>
<td>![multiple_icon]</td>
</tr>
</tbody>
</table>
Business Process Modelling Notation (BPMN)

Extended Element Set (e.g. Event types)

<table>
<thead>
<tr>
<th>“Catching”</th>
<th>“Throwing”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>Message</td>
</tr>
<tr>
<td>Timer</td>
<td>Timer</td>
</tr>
<tr>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>Cancel</td>
<td>Cancel</td>
</tr>
<tr>
<td>Compensation</td>
<td>Compensation</td>
</tr>
<tr>
<td>Conditional</td>
<td>Conditional</td>
</tr>
<tr>
<td>Link</td>
<td>Link</td>
</tr>
</tbody>
</table>

**ErrorCode attribute for Error Event**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorCode : String</td>
<td><em>For an End Event:</em> If the Result is an Error, then the ErrorCode MUST be supplied. This “throws” the error. [...]</td>
</tr>
</tbody>
</table>
Our Contribution:

An ontology for BPMN

- An OWL-DL formalization of the BPMN specification
- It accurately encodes:
  - the classification of all the elements of the BPMN language
  - the formal representation of the attributes and conditions describing how the elements can be combined to obtain a “valid” BPMN business process
- The proposed formalization:
  - provides a terminological description of the language;
  - enables representing any actual BPMN diagram as a DL A-Box
    * enables several reasoning-based services
- It covers BPMN v1.1...
  - we are in the process of updating to BPMN 2.02
Disclaimer

• The BPMN Ontology…
  - …is not intended to model the dynamic behaviour (behavioural semantics) of a BPMN process
    • Ontology languages are not particularly suited for this
    • better look at YAWL, PetriNets, …
  - …it provides an ontological formalization of BPMN as a graphical language, and not an ontological analysis in a foundational fashion
    • better look at works analysing BPMN wrt to
      - ABDESO/UFO (Guizzardi and Wagner)
      - Dolce (Sanfilippo, Borgo, and Masolo - see conference next session)
Modelling Process
Scope and Boundaries: Ontology Intended Uses

• Checking the **compliance** of a process diagram against the **BPMN specification**
  - e.g., the process diagram has at least one starting event and one end event, constructs are combined in the correct way

• Checking additional **application-specific design guidelines**
  - guidelines to guarantee process diagram readability (e.g., diagram should not contain more than ten sub-processes, every gate should have at most three out-going flows)

• **Semantic description and retrieval** of process diagrams (or process diagram elements)
  - e.g., to state that a certain sub-process is of type “privacy critical”, and to be able to retrieve all process diagrams that contains privacy critical sub-processes, or all privacy critical activities within a diagram
Modelling Process
Scope and Boundaries: Competency Questions (excerpt)

• 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?
• ....
Modelling Process
Our Trusted Friend: BPMN Specification Document

• For each element, it provides:
  - an introductory description of the element, with some general properties and conditions
  - a compact tabular description of each element’s attribute
    • name, value type, multiplicity details, conditions for instantiation
  - conditions holding for connecting the current element with other elements of the language
  - additional details on execution level aspects of the element

• Free text document, with some structure
Modelling Process
Step 1 of 3: Signature Identification

• Identification and classification of all the elements of the language
  - We associated each element of the language to a class in the ontology
  - Guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology
Modelling Process
Step 1 of 3: Signature Identification

- standard loop a.
- multi instance loop a.
- embedded s.
- reference s.
- reusable s.
- abstract t.
- manual t.
- receive t.
- reference t.
- task
- script t.
- send t.
- service t.
- user t.
- multiple i.e.
- conditional i.e.
- compensation i.e.
- none i.e.
- link i.e.
- error i.e.
- not activity boundary i.e.
- message i.e.
- start e.
- end e.
- event
- intermediate e.
- activity
- flow object
- gateway
- connecting object
- sequence flow
- message flow
- association
- parallel g.
- inclusive g.
- complex g.
- exclusive g.
- event based e.g.
- data based e.g.
- data object
- annotation
- group
- swimlane
- activity
- graphical element
- pool
- lane
• Identification and classification of all the elements of the language
  - We associated each element of the language to a class in the ontology.
  - Guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology.

Modelling Process
Step 1 of 3: Signature Identification
Modelling Process
Step 1 of 3: Signature Identification

An ontology for the Business Process Modelling Notation - Rospocher at al.
Identification and classification of all the elements of the language
- We associated each element of the language to a class in the ontology.
- Guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology.

Modelling Process
Step 1 of 3: Signature Identification
Identification and classification of all the elements of the language

- We associated each element of the language to a class in the ontology.
- Guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology.
Modelling Process
Step 2 of 3: Attribute Restrictions

• An attribute is formalized either as **datatype** property or as an **object** property

• Three situations considered:
  1. the value type of the attribute is another BPMN element
  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
  3. the value type of the attribute is a datatype with no restriction
Modelling Process
Step 2 of 3: Attribute Restrictions (cont’d)

• Case I: The value type of the attribute is another BPMN element

• Example:
  - Target attribute of Intermediate Event [p47]

| Target (0-1) : Activity | A Target MAY be included for the Intermediate Event. The Target MUST be an activity (Sub-Process or Task). This means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity. |

• Formalization: as object property
  - domain: the class having the attribute
  - range: the class of the element mentioned as value type of the attribute

\[
\exists \text{hasIntermediateEventTarget}. \top \sqsubseteq \text{IntermediateEvent} \\
\top \sqsubseteq \forall \text{hasIntermediateEventTarget}. \text{Activity}
\]
Case II: 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.

Example:
- AdHocOrdering attribute of Embedded SubProcess [p47]

Formalization: as object property
- domain: the class having the attribute
- range: a new class enumerating all possible values of the attribute

\[ \exists \text{hasESPAdHocOrdering}. \top \sqsubseteq \text{EmbeddedSubProcess} \]
\[ \top \sqsubseteq \forall \text{hasESPAdHocOrdering} \cdot \text{AdHocOrderingType} \]
Modelling Process
Step 2 of 3: Attribute Restrictions (cont’d)

• Case III: The value type of the attribute is a datatype with no restriction

• Example:
  - Text attribute of Text Annotation [p95]

| Text : String       | Text is an attribute which is text that the modeler wishes to communicate to the reader of the Diagram. |

• Formalization: as datatype property
  - domain: the class having the attribute
  - range: a datatype compatible with the value type of the attribute

\[
\exists \text{hasTextAnnotationText}. T \subseteq \text{TextAnnotation} \\
T \subseteq \forall \text{hasTextAnnotationText}. \text{DT}\{\text{string}\}
\]
For each attribute, we formalized its multiplicity details as an OWL cardinality restriction on the class having the attribute:

- (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
- (0..n) multiplicity is not encoded at all

Example:
- State attribute of Data Object [p94]

<table>
<thead>
<tr>
<th>State (0-1) : String</th>
</tr>
</thead>
<tbody>
<tr>
<td>State is an optional attribute that indicates the impact the Process has had on the Data Object. Multiple Data Objects with the same name MAY share the same state within one Process.</td>
</tr>
</tbody>
</table>

\[
\text{DataObject} \sqsubseteq (\leq 1)\text{hasState}
\]
For each attribute, we also encode **additional conditions** ruling the usage of the attribute.

**Example:**
- ErrorCode attribute of Error, in case the Error is a result of an End Event (p.94)

<table>
<thead>
<tr>
<th>ErrorCode: String</th>
</tr>
</thead>
<tbody>
<tr>
<td>For an End Event:</td>
</tr>
<tr>
<td>If the Result is an Error, then the ErrorCode MUST be supplied. This “throws” the error.</td>
</tr>
</tbody>
</table>

**Formalization:** case by case

\[
\text{EndEvent} \subseteq \neg \exists \text{hasResult.Error} \quad \exists \text{hasResult.}(\text{Error} \cap \exists \text{hasErrorCode})
\]
Modelling Process
Step 3 of 3: Structural Constraints Formalization

• Formalization of the **conditions** concerning the **usage** of the elements of the language to compose a **BPMN diagram**

• Examples: [p48,p72]

  - **A Start Event MUST be a source for Sequence Flow.**

| Gates (0-n) : Gate | There MAY be zero or more Gates (except where noted below). Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates. |

• Formalization: **case by case**
A Start Event MUST be a source for Sequence Flow.

\[ \text{StartEvent} \sqsubseteq \exists \text{hasConnectingObjectSource}^{-1}.\text{SequenceFlow} \]

<table>
<thead>
<tr>
<th>Gates ((0-n)): Gate</th>
<th>There MAY be zero or more Gates (except where noted below). Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates.</th>
</tr>
</thead>
</table>

\[ \text{Gateway} \sqsubseteq (\geq 2)\text{hasSequenceFlowTarget}^{-1} \sqcup \]
\[ ((\leq 1)\text{hasSequenceFlowTarget}^{-1} \sqcap (\geq 2)\text{hasGatewayGate}) \]
The BPMN Ontology

Limitations

- A few documented properties and conditions are not encoded in the BPMN Ontology:
  - Execution level properties
  - Attribute default values
  - “Undecideable” conditions
The BPMN Ontology
Limitations: Execution level properties

• These properties specify the **behavioral nature** of the **graphical elements** in a BPMN diagram.

• Example: [p.77]

To define the exclusive nature of this Gateway’s behavior for diverging Sequence Flow:

- If there are multiple outgoing Sequence Flow, then only one Gate (or the DefaultGate) SHALL be selected during performance of the Process.

• Not considered, as our BPMN Ontology is not intended to model the dynamic behaviour of BPMN diagrams.
The BPMN Ontology

Limitations: Attribute default values

- **Attributes default values**: the value taken by the required attributes when not explicitly assigned in the model

- Example: [p.52]

<table>
<thead>
<tr>
<th>Implementation (Web Service</th>
<th>Other</th>
<th>Unspecified) Web Service : String</th>
</tr>
</thead>
<tbody>
<tr>
<td>This attribute specifies the technology that will be used to send or receive the message. <strong>A Web service is the default technology.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Attributes default values have not been formalized because OWL does not support the specification of properties default values
The BPMN Ontology

Limitations: “Undecideable” conditions

- There is a limited number of conditions which are not represented
  - \(\neg p \neg q \neg r\)
  - We wanted to remain in a decidable fragment of OWL

- Example: [p73]

<table>
<thead>
<tr>
<th>OutgoingSequenceFlow : SequenceFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each Gate MUST have an associated (outgoing) Sequence Flow. The attributes of The Sequence Flow MUST have its Condition attribute set to Expression and MUST have a valid ConditionExpression. The ConditionExpression MUST be unique for all the Gates within the Gateway. If there is only one Gate (i.e., the Gateway is acting only as a Merge), then Sequence Flow MUST have its Condition set to None.</td>
</tr>
</tbody>
</table>

- This condition could be formalized by defining a functional property as the chain of two other properties
  - One connecting gateways to sequence flows
  - One associating conditions to sequence flows

But imposing number restrictions on property chains (or, more generally, on complex roles) lead to undecidability
The BPMN Ontology

Ontology Metrics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL Expressivity</td>
<td>$\mathcal{SHOIN}(D)$</td>
</tr>
<tr>
<td>Classes</td>
<td>117</td>
</tr>
<tr>
<td>Object Properties</td>
<td>123</td>
</tr>
<tr>
<td>Datatype Properties</td>
<td>48</td>
</tr>
<tr>
<td>Individuals</td>
<td>104</td>
</tr>
<tr>
<td>Class Axioms</td>
<td>463</td>
</tr>
<tr>
<td>Object Property Axioms</td>
<td>236</td>
</tr>
<tr>
<td>Datatype Property Axioms</td>
<td>96</td>
</tr>
<tr>
<td>Individual Axioms</td>
<td>250</td>
</tr>
<tr>
<td>Annotation</td>
<td>504</td>
</tr>
</tbody>
</table>
Instantiating the BPMN Ontology

- Given a BPMN business process diagram (BPD), it is possible to represent it as an A-box in the language of the BPMN Ontology.

- Example:
Instantiating the BPMN Ontology

Step 1. Create an individual for each element

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 tasks
Instantiating the BPMN Ontology

Step 1. Create an individual for each element

BPD individuals

- $p_1$ 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
Instantiating the BPMN Ontology

Step 2. Instantiate each individual wrt its BPMN class

**Type assertions**

- `embedded_loop_sub_process(p₁)`
- `data_based_exclusive_gateway(gᵢ)` where `i = 1, 2`
- `sequence_flow(sⱼ)` where `j = 1, .., 4`
- `task(tₖ)` where `k = 1, 2`
Instantiating the BPMN Ontology

Step 2. Instantiate each individual wrt its BPMN class

Type assertions

<table>
<thead>
<tr>
<th>Type assertion</th>
<th>i</th>
<th>j</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>embedded_loop_sub_process($p_1$)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data_based_exclusive_gateway($g_i$)</td>
<td>$i = 1, 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence_flow($s_j$)</td>
<td>$j = 1, \ldots, 4$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>task($t_k$)</td>
<td></td>
<td></td>
<td>$k = 1, 2$</td>
</tr>
</tbody>
</table>
Instantiating the BPMN Ontology

Step 3. “Compose” the diagram structure in the A-box

```
Structural assertions

- has_graphical_elements(p₁, sᵢ) \( j = 1, .., 4 \)
- has_graphical_elements(p₁, gᵢ) \( i = 1, 2 \)
- has_graphical_elements(p₁, tᵢ) \( k = 1, 2 \)
- has_sequence_flow_source_ref(s₁, g₁)
- has_sequence_flow_target_ref(s₁, t₁)
- has_sequence_flow_source_ref(s₂, g₂)
```
Instantiating the BPMN Ontology

Step 3. “Compose” the diagram structure in the A-box.

Structural assertions:

- \( \text{has_graphical_elements}(p_1, s_j) \), \( j = 1, \ldots, 4 \)
- \( \text{has_graphical_elements}(p_1, g_i) \), \( i = 1, 2 \)
- \( \text{has_graphical_elements}(p_1, t_k) \), \( k = 1, 2 \)
- \( \text{has_sequence_flow_source_ref}(s_1, g_1) \)
- \( \text{has_sequence_flow_target_ref}(s_1, t_1) \)
Instantiating the BPMN Ontology

Step 3. “Compose” the diagram structure in the A-box
Instantiating the BPMN Ontology

Step 3. “Compose” the diagram structure in the A-box

[Diagram of a fully-structured Access Mode: Hospital administration]
Instantiating the BPMN Ontology
Reasoning over an instantiated BPMN Ontology

• **Query answering** on BPMN diagrams (via SPARQL)
  - "Which are the activities which follows gateways and produce a data object?"
  - "Are there sub-processes which do not contain start/end events?"

• **Compliance checking** of a BPMN diagram against the BPMN Specification
  - e.g.: $\text{Gateway} \sqsubseteq (\geq 2)\text{hasSequenceFlowTarget}^{-1} \sqsupseteq$
    
    
    $((\leq 1)\text{hasSequenceFlowTarget}^{-1} \sqsubseteq$

    
    

    $\geq 2)\text{hasGatewayGate}$

  - doable, but in closed-world assumption!
Application Scenarios

• In (Semantic Web) applications where a terminological description of BPMN is needed
  - e.g.: for information retrieval of resources annotated with respect to the elements of the language

• To favour knowledge interoperability (via ontology mappings)
  - e.g.: in a product lifecycle management environment

• To check the compliance of a BPD against the BPMN Specification
  - extendable to domain/application specific constraints
    - e.g., keep the model as simple and readable as possible
    - Gateways in the process may have at most two outgoing sequence flows
    - e.g., ensure that exceptional situations are always handled
    - 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
Application Scenarios
Reasoning on semantically annotated business processes
Application Scenarios

Reasoning on semantically annotated business processes

- Semantically annotated business processes can be encoded into a OWL knowledge base
Application Scenarios
Reasoning on semantically annotated business processes

• The framework also **enables to define constraints for**
  - **correct/incorrect annotation** of business process
    • e.g.: A BPMN activity is annotatable only with actions of the domain ontology (and not e.g., with documents)
  - **valid critical patterns**
    • e.g.: the activity of reserving products in the On-line Shop pool has always to catch a “product unavailability” error event

• Using DL-reasoning we can
  - Check **compatibility of process constraints**
  - **Verify constraints** over an annotated process
  - **Perform queries combining** domain and BPMN semantics
Conclusions

- We presented an ontological (OWL-DL) formalization of the BPMN specification

- The proposed formalization:
  - provides a terminological description of the language;
  - enables representing any actual BPMN diagram as a set of individuals and assertions on them
    - enables several reasoning-based services

- Available on-line:
  - dkm.fbk.eu (~25/30 download per month)
  - ontohub.org

- Current / future work:
  - cover BPMN 2.02
Thank you! Questions?

Marco Rospocher
Fondazione Bruno Kessler, Data and Knowledge Management Unit
Trento, Italy
rospocher@fbk.eu :: http://dkm.fbk.eu/rospocher