The business process modelling ontology
The Business Process Modelling Ontology

Liliana Cabral
KMi, The Open University
Milton Keynes, UK
+44 1908 653800
l.s.cabral@open.ac.uk

Barry Norton
KMi, The Open University
Milton Keynes, UK
+44 1908 653800
b.j.norton@open.ac.uk

John Domingue
KMi, The Open University
Milton Keynes, UK
+44 1908 653800
j.b.domingue@open.ac.uk

ABSTRACT
In this paper we describe the Business Process Modelling Ontology (BPMO), which is part of an approach to modelling business processes at the semantic level, integrating knowledge about the organisational context, workflow activities and Semantic Web Services. We harness knowledge representation and reasoning techniques so that business process workflows can: be exposed and shared through semantic descriptions; refer to semantically annotated data and services; incorporate heterogeneous data though semantic mappings; and be queried using a reasoner or inference engine. In this paper we describe our approach and evaluate BPMO through a use case.

Categories and Subject Descriptors
I.2.4 [Knowledge Representation Formalisms and Methods]: Representations. F.3.2 [Semantics of Programming Languages]: Process models.

General Terms
Design, Standardization, Management, Languages.

Keywords

1. INTRODUCTION
Business organisations today need agility and flexibility to deal with highly dynamic environments, providing ever-changing services and products as well as in interacting with diverse customers and partners. A significant problem in the area of Business Process Management (BPM) lies in bridging between the organisational context, the diverse process workflow notations, and the executable services that fulfill process activities, by which business analysts would like to understand, maintain and adapt their business processes.

Currently, business analysts use process modelling notations such as BPMN [11] and EPC [14] to define business process models as part of tool suites for BPM. These notations are useful at the business level, but alone they provide no inference reasoning over business processes. For example, BPMN tools are rich in control-flow constructs, but the graphical elements contain only limited textual information with no formal semantics. Some EPC-based tools such as ARIS [14] on the other hand, provide integration of different views (e.g. organisation, data and control) and levels (e.g. requirements and implementation); however mediating between these views and levels is a very complex task due to the variety of underlying representations. In addition, it is very difficult to use these notations to automatically query the business context or draw relations between existing processes or services.

In this paper we present the Business Process Modelling Ontology\(^1\) (BPMO), which plays a key role in solving the problem above, by enabling the semantic annotation of high-level business process models, which we assume can be fulfilled by Web services. This work is part of the SUPER project\(^2\), which in particular provides a set of integrated ontologies for facilitating Semantic Business Process Management [6].

The rest of this paper is structured as follows. Section 2 describes the BPMO approach. Section 3 describes the main concepts of BPMO. Section 4 illustrates the use of BPMO and shows results through a use case. Section 5 presents our conclusions and related work.

2. APPROACH OVERVIEW
The Business Process Modelling Ontology (BPMO) is part of an approach to modelling business processes at the semantic level, integrating knowledge about the organisational context, workflow activities and Semantic Web Services. This approach provides support for various BPM activities, from modelling and querying to execution and analysis; regardless of specific notations in a manner which crosses domains and organisational boundaries. We harness a number of knowledge representation and reasoning techniques so that business process workflows can: be exposed and shared through semantic descriptions; refer to semantically annotated business data and services; incorporate heterogeneous data though semantic mappings; and be queried using a reasoner or inference engine. We argue therefore, that BPMO enables

\(^1\) http://www.ip-super.org/ontologies/process/bpmo/v2.0.1#bpmo
\(^2\) Semantics Utilised for Process Management within and between Enterprises (http://www.ip-super.org)
seamless interoperation, querying, sharing, mediation and translation of business processes.

Within our approach, a business analyst can draw a business process diagram with a tool that automatically generates BPMO instances (see example in Section 3), or he can use translators that will transform specific notations from and to BPMO. BPMO can thus be viewed as a bridging ontology enabling the annotation of business processes workflows extended with organisational context and automated translation between an open-ended set of existing notations and languages.

As we will show in the rest of the paper, the BPMO model captures domain independent organisational aspects, control-flow features of notations such as BPMN, via a number of workflow patterns as in [1], process interaction features from BPEL\(^3\), and finally service description and invocation features from Semantic Web Services (SWS) [4]. BPMO builds on the formalization of Business Process Diagrams as presented in [12], and as such is oriented towards the production of well-formed workflow models, where graphs decompose unambiguously into sub-graphs that start and end with compatible constructs.

More specifically, in the SUPER project we provide ontologies for standards such as BPMN, EPC and BPEL (see [2], [5], [10]) as well as corresponding translators to BPMO. In addition, it is possible for business analysts to create alternative organisational ontologies to define BPMO process organisational attributes. This is done via UPO\(^4\) (Upper-level Process Ontology), an ontology defining high-level business process concepts, which are shared by all ontologies in SUPER.

A BPMO diagram can be defined using the WSMO Studio BPMO Modeller tool\(^5\), which automatically generates instances of BPMO in WSML [16]. More precisely, we use WSML-Flight, which adds F-Logic like features to the WSML core, directly supporting WSMO Web Service descriptions [4]. WSML-Flight also allows us to apply data mappings (via rule-type axioms) directly in the ontology language without having to rely on a hybrid approach of a separate rule language. BPMO and the related ontologies mentioned above are publicly available at the SUPER website (http://www.ip-super.org/ontologies).

### 3. BPMO DESCRIPTION

BPMO is a representation for high-level business process workflow models, abstracting from existing business process notations. Nevertheless, the workflow elements of a BPMO process diagram comply with a corresponding subset of BPMN control-flow elements and are informed by, and named according to Workflow Patterns [1]. Moreover, BPMO concepts related to interaction activities (e.g. Send, Receive) have a number of attributes that correspond to BPEL constructs.

Basically, a BPMO process description captures the business context of the modelled process and contains the process workflow, which represents the behaviour of the process (through control-flow and data-flow constructs) and process activities (through Tasks). The main BPMO process elements are structured as follows (see Table 1):

- **Workflow** – The business process container for Workflow Elements. The initial Workflow Element may be a StartEvent or a block pattern, commonly a Sequence or ParallelSplit Synchronise. If the StartEvent is present, subsequent elements will be linked in graph style by ControlFlow Connectors. If the Workflow Element is a Sequence, a sequence flow is implicit between the contained elements. If the Workflow Element is ParallelSplitSynchronise, a parallel flow is implicit.

- **Workflow Elements** – These are general elements that belong to a business process workflow, including Processes, Tasks, Events, block patterns and graph patterns;

- **Block Patterns** – These are structured or pattern-based control-flow elements representing workflow decision points (gateways), including Sequence, ParallelSplitSynchronise, ExclusiveChoiceMerge, DeferredChoiceMerge, While, Repeat, and so on.

- **Graph patterns** – These are link based control-flow elements representing workflow decision points (gateways), including ParallelSplit, ExclusiveChoice, DeferredChoice, SimpleMerge, Synchronise, and so on.

As can be seen, BPMO combines features of block-oriented and graph-oriented workflow models. The main purpose of block patterns is to explicitly represent structured elements and workflow patterns that can be used to facilitate process verification and the translation to notations in the execution level. The BPMO design enforces well formed diagrams, via graph patterns and structures, but further restrictions can be easily provided via axioms.

<table>
<thead>
<tr>
<th>BPMO Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartEvent</td>
<td>An event signalling the start of a process</td>
</tr>
<tr>
<td>TimerEvent</td>
<td>An event signalling that a specific time has been reached</td>
</tr>
<tr>
<td>ErrorEvent</td>
<td>An event signalling that an error has occurred</td>
</tr>
<tr>
<td>EndEvent</td>
<td>An event signalling the end of a process</td>
</tr>
<tr>
<td>Task</td>
<td>An atomic activity within a Process</td>
</tr>
<tr>
<td>Goal Task</td>
<td>A Task with an attached Semantic Capability, used for invoking SWS goals</td>
</tr>
<tr>
<td>Receive</td>
<td>A Task for receiving messages. Provides a semantic description of the requested capability.</td>
</tr>
<tr>
<td>ReceiveMessageEvent</td>
<td>A Receive task associated with an event (which may resolve choices, see DeferredChoice), and so on.</td>
</tr>
<tr>
<td>Mediation Task</td>
<td>A Task for dataflow and data mediation</td>
</tr>
<tr>
<td>Sequence</td>
<td>An ordered set of activities (also a linked list) with an implicit sequence flow (Block pattern)</td>
</tr>
<tr>
<td>ParallelSplit</td>
<td>A gateway (or decision point) for creating concurrent branches</td>
</tr>
<tr>
<td>Synchronisation</td>
<td>A gateway for synchronizing concurrent branches</td>
</tr>
<tr>
<td>ExclusiveChoice</td>
<td>A decision gateway for selecting one out of a set of mutually exclusive alternative branches based on</td>
</tr>
</tbody>
</table>

---

\(^3\) [http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf](http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf)

\(^4\) [http://www.ip-super.org/ontologies/process/upo/v2.0.1#upo](http://www.ip-super.org/ontologies/process/upo/v2.0.1#upo)

The concepts related to interaction tasks in BPMO are GoalTask, Receive, Send and ReceiveMessageEvent (see Listing 2), which are subconcepts of Task. A Task is also a Business Activity (as in Listing 1) and thus can also refer to business attributes such as a business policy or a business process goal. Tasks have attributes to represent information about the interaction with a partner process, such as partner role (hasPartnerRole), inputs (hasInputDescription) and outputs (hasOutputDescription). Most attribute types in Tasks are defined as SemanticCapability which is a wrapper for ontology elements or service descriptions. For example, a SemanticCapability instance can refer to the URI of an concept within an ontology or to the URI of a Web Service or Goal description.

<table>
<thead>
<tr>
<th>BPMO Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExclusiveChoiceMerge</td>
<td>Exclusive Choice with an implicit Simple Merge (Block pattern)</td>
</tr>
<tr>
<td>DeferredChoice</td>
<td>A deferred choice gateway for selecting one out of the set of mutually exclusive alternative branches based on external event</td>
</tr>
<tr>
<td>DeferredChoiceMerge</td>
<td>Deferred Choice with an implicit Simple Merge (Block pattern)</td>
</tr>
<tr>
<td>SimpleMerge</td>
<td>Gateway for joining a set of mutually exclusive alternative branches into one branch</td>
</tr>
<tr>
<td>MultipleChoice</td>
<td>A decision gateway for selecting many out of a set of alternative branches into several parallel branches based on data. One of the branches may be default.</td>
</tr>
<tr>
<td>MultipleMerge</td>
<td>Unsynchronised convergence of two or more distinct branches</td>
</tr>
<tr>
<td>MultipleChoiceMerge</td>
<td>Multiple choice with an implicit multiple merge (Block pattern)</td>
</tr>
<tr>
<td>MultipleMergeSynchro nise</td>
<td>Synchronised convergence of two or more distinct branches</td>
</tr>
<tr>
<td>Repeat</td>
<td>A structured loop where the condition is evaluated after the body of the loop is executed</td>
</tr>
<tr>
<td>While</td>
<td>A structured loop where the condition is evaluated before the body of the loop is executed</td>
</tr>
</tbody>
</table>

We will discuss next the use of a number of key BPMO concepts, which are defined in WSML, including Process, Business Activity, Task (Send, Receive, GoalTask), SemanticCapability, MediationTask and DataMediator.

The Process concept (shown in Listing 1) defines several organisational attributes, by inheriting from BusinessActivity, according to the types BusinessDomain, BusinessFunction, BusinessStrategy, BusinessPolicy, BusinessProcessMetrics, BusinessProcessGoal and BusinessResource. These business-level concepts (attribute types) are primarily defined in external ontologies, which model a specific business domain and organisation. These ontologies are linked to the BPMO process by subclassing the UPO concept (note that upo# is the prefix for the UPO namespace). As a result, we enable the querying of processes against organisational aspects by business analysts (see example in the next section). The Process itself can also have a corresponding Web Service description (hasWSDescription attribute). In addition, the Process concept defines the process workflow (attribute hasWorkflow). The concept Workflow defines the first element of the workflow (hasFirstWorkflowElement). The workflow is modelled with Workflow Elements contained in or following (via connectors) the first element.

<table>
<thead>
<tr>
<th>Concept</th>
<th>BusinessActivity subConceptOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasName ofType</td>
<td>string</td>
</tr>
<tr>
<td>hasDescription ofType</td>
<td>string</td>
</tr>
<tr>
<td>hasNonFunctionalProperties ofType</td>
<td>BusinessActivityNonFunctionalProperties</td>
</tr>
<tr>
<td>hasBusinessDomain ofType</td>
<td>upo#BusinessDomain</td>
</tr>
<tr>
<td>hasBusinessFunction ofType</td>
<td>upo#BusinessFunction</td>
</tr>
<tr>
<td>hasBusinessStrategy ofType</td>
<td>upo#BusinessStrategy</td>
</tr>
<tr>
<td>hasBusinessPolicy ofType</td>
<td>upo#BusinessPolicy</td>
</tr>
<tr>
<td>hasBusinessProcessMetrics ofType</td>
<td>BusinessProcessMetrics</td>
</tr>
<tr>
<td>hasBusinessGoal ofType</td>
<td>upo#BusinessProcessGoal</td>
</tr>
<tr>
<td>hasBusinessResource ofType</td>
<td>upo#BusinessResource</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>GoalTask subConceptOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasPartnerGoal ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>hasPartnerRole ofType</td>
<td>BusinessRole</td>
</tr>
<tr>
<td>messageTo ofType</td>
<td>Receive</td>
</tr>
<tr>
<td>messageFrom ofType</td>
<td>Send</td>
</tr>
<tr>
<td>hasInputDescription ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>hasOutputDescription ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>requestsCapability ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>providesCapability ofType</td>
<td>SemanticCapability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>Send subConceptOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasPartnerWebService ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>hasPartnerRole ofType</td>
<td>BusinessRole</td>
</tr>
<tr>
<td>hasReceiveCounterpart ofType</td>
<td>Receive</td>
</tr>
<tr>
<td>hasInputDescription ofType</td>
<td>SemanticCapability</td>
</tr>
<tr>
<td>hasOutputDescription ofType</td>
<td>SemanticCapability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>ReceiveMessageEvent subConceptOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasPartnerWebService ofType</td>
<td>BusinessRole</td>
</tr>
<tr>
<td>hasSendCounterpart ofType</td>
<td>Send</td>
</tr>
<tr>
<td>hasInputDescription ofType</td>
<td>SemanticCapability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>ReceiveTask subConceptOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNonFunctionalProperties</td>
<td>string</td>
</tr>
<tr>
<td>hasBusinessDomain ofType</td>
<td>upo#BusinessDomain</td>
</tr>
<tr>
<td>hasBusinessFunction ofType</td>
<td>upo#BusinessFunction</td>
</tr>
<tr>
<td>hasBusinessStrategy ofType</td>
<td>upo#BusinessStrategy</td>
</tr>
<tr>
<td>hasBusinessPolicy ofType</td>
<td>upo#BusinessPolicy</td>
</tr>
<tr>
<td>hasBusinessProcessMetrics ofType</td>
<td>BusinessProcessMetrics</td>
</tr>
<tr>
<td>hasBusinessGoal ofType</td>
<td>upo#BusinessProcessGoal</td>
</tr>
<tr>
<td>hasBusinessResource ofType</td>
<td>upo#BusinessResource</td>
</tr>
</tbody>
</table>

A GoalTask represents an atomic activity, which can be automatically achieved through a SWS invocation (synchronous communication). The attribute hasPartnerGoal is used in this case to refer to the Goal description. The hasInputDescription and hasOutputDescription attributes refer to the semantic descriptions of request and response data respectively. Hence, dataflow is enabled by sharing the same data description across tasks in the workflow. The requestsCapability and providesCapability attributes refer to the semantic descriptions of operations related to request and response respectively. The Send and Receive tasks are similar to Goal tasks, but they are used for asynchronous communication. A Receive task can be associated with a Send in the same workflow via the hasSendCounterpart attribute (and conversely for Send). ReceiveMessageEvent works as a Receive task, but is also associated to an event, which is triggered when a message is received.
Listing 3. Concepts related to Mediation

<table>
<thead>
<tr>
<th>Concept</th>
<th>SubConceptOf</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>MediationTask</td>
<td>subConceptOf</td>
<td>Task</td>
<td>hasSourceTask, hasTargetTask, hasDataMediator</td>
</tr>
<tr>
<td>Mediator</td>
<td>subConceptOf</td>
<td>upo#businessProcessMediator</td>
<td>hasName, hasDescription</td>
</tr>
<tr>
<td>ProcessMediator</td>
<td>subConceptOf</td>
<td>Mediator</td>
<td>hasSourceProcess, hasTargetProcess, hasMediationProcess</td>
</tr>
<tr>
<td>DataMediator</td>
<td>subConceptOf</td>
<td>Mediator</td>
<td>hasMediationService, hasInputDescription, hasOutputDescription</td>
</tr>
<tr>
<td>MediationProcess</td>
<td>subConceptOf</td>
<td>Process</td>
<td></td>
</tr>
</tbody>
</table>

BPMO also supports data and process mediation through a number of concepts as shown in Listing 3. See also the examples in the next section. A MediationTask is a task that provides data mapping specifications to be used between tasks during runtime. A MediatorTask can have one or more DataMediators. The DataMediator concept refers to a data mediator or mediation service and the input and output for them. In a typical use case, the hasMediator attribute will refer to a mapping definition (URI), the hasInputDescription will refer to a source ontology (URI) and the hasOutputDescription will refer to a target ontology (URI). In addition, the ProcessMediator concept is used as a descriptor to identify a process with a mediation role (hasMediationProcess) and mediated processes (hasSourceProcess, hasTargetProcess) in order to facilitate the job of tools for verification and creation of mediation processes. The ProcessMediator can also refer to a mediator component (hasSWSMediator).

4. USE CASE

In this section we will illustrate and evaluate the BPMO model through an example taken from the mediation scenario provided in the SWS Challenge (http://sws-challenge.org). This scenario is about a Purchase Order process and involves three partners: the service requester (customer), company Moon, which orders products; and the mediator, which must be implemented to mediate between Blue and Moon. The goal of the mediator is to map the incoming and outgoing messages between Blue and Moon and also invoke required services from Moon so that the interactions necessary to buy a product are complete. Company Blue sends a purchase order and receives an acknowledgment via the mediator. In this paper we focus on the part of the solution comprising the use of ontology based-technology, required to solve the scenario above. We use BPMO for modelling the main process using SWS references for Task descriptions, and domain ontologies for modelling data and mappings.

We created a BPMO diagram to represent the mediator process (Moon Mediator Process) as shown in Figure 1, using WSMO studio’s BPMO modeller. The modeller generates an initial set of BPMO instances corresponding to the process control-flow, to which the user can add data instances for attributes using the modeller’s property editor. A number of BPMO instances corresponding to the diagram presented in Figure 1 are shown in Listing 4.

![Figure 1. BPMO diagram of the Mediator process](image-url)

The Moon Mediator Process workflow is quite self-explanatory. Basically, we used activities Receive Purchase Order and Send PO Confirmation to interact with the Blue Company, modelled as Receive and Send BPMO tasks accordingly. Map Purchase Order and Map Result are modelled as Mediation Tasks and used to map the values needed by the Moon and Blue Web Services, respectively. Search Customer, Create Order and Close Order are modelled as Goal Tasks for invoking Web Services provided by Moon. Add Line Item and Confirm Line Item are modelled as corresponding asynchronous Send and Receive tasks, which are called in a Repeat (loop) for every item in the received Purchase Order. Start and End are events used to initiate and finalise the process.

Listing 4. BPMO instances for the Mediator Process

```
instance Process_MoonMediator memberOf bpmo#Process
bpmo#hasName hasValue "Moon Mediator Process"
bpmo#hasWorkflow hasValue Workflow_1217
bpmo#hasBusinessStrategy hasValue moon#strategy45
bpmo#hasBusinessPolicy hasValue moon#policy33
bpmo#hasOrganization hasValue kmi

instance kmi memberOf upo#Organization
upo#hasName hasValue "KMI, The Open University, UK"

instance Blue memberOf upo#Organization
upo#hasName hasValue "Blue, SWS Challenge"

instance customer memberOf bpmo#Organization
bpmo#hasName hasValue "Blue Customer"
bpmo#hasOrganization hasValue KMI

instance moon memberOf upo#Organization
upo#hasName hasValue "Moon, SWS Challenge"

instance moonCRM memberOf bpmo#Organization
bpmo#hasName hasValue "Moon Customer Relationship Management"
bpmo#hasOrganization hasValue Moon

instance Workflow_1217 memberOf bpmo#Workflow
bpmo#hasHomeProcess hasValue Process_MoonMediator
bpmo#hasFirstWorkflowElement hasValue StartEvent_1

instance StartEvent_1 memberOf bpmo#StartEvent
bpmo#hasHomeProcess hasValue Process_MoonMediator
bpmo#hasName hasValue "Start"

instance ControlflowConnector_100 memberOf
bpmo#ControlflowConnector
bpmo#hasHomeProcess hasValue Process_MoonMediator
```
In this example starting with a Start event, all workflow elements are linked sequentially in an explicit way using ControlflowConnector. Each ControlflowConnector points to a source WorkflowElement and a target WorkflowElement. For example, StartEvent_1 is linked to Receive task Receive.ReceivePO via ControlflowConnector_100. Note that in this example we use structured loops (Repeat), which are treated as one block element (with a condition and an execution body).

Note also in Listing 4 how we have added information about the partner roles and involved organisations. The attributes hasBusinessStrategy and hasBusinessPolicy have values that refer to strategies and policies defined by Moon Company in an external ontology. The value of attribute apio#hasInvolvedRole in Process_MoonMediator defines the role (BusinessRole) of this process. In this case kmi (Organisation) is playing the mediator role (moon Mediator). In a similar way we define the roles of blue (customer) and moon (moonCRM) partners. In Receive.ReceivePO we provide values for attributes hasPartnerRole, hasPartnerWebService, hasSendCounterpart and hasInputDescription. These attributes values are necessary in order to establish an interaction with a partner. In particular, Receive.ReceivePO obtains data from partner blue. The data received is defined in hasInputDescription (via SemanticCapability), which in this case kmi.open.ac.uk/swsc/datamediator/PurchaseOrderRequest. The definition of PurchaseOrderRequest (omitting namespace) and other concepts used in the scenario are shown in Listing 5 together with some instances. These concepts and instances have been derived from the XML Schema given in the original scenario, but simplified here for illustration purposes.

### Listing 5 Domain data ontology concepts and instances

```
<instance bpmo#Receive_ReceivePO memberOf bpmo#Receive
  bpmo#hasName hasValue "Receive Purchase Order"
  bpmo#hasPartnerWebService hasValue MoonMediator
  SemanticCapability ReceivePO_WSMO
  bpmo#hasPartnerRole hasValue customer
  bpmo#hasSendCounterpart hasValue Send_SendPOConf
  bpmo#hasInputChange hasValue SemanticCapability_PurchaseOrderDesc
  instance SemanticCapability_PurchaseOrderDesc memberOf
  bpmo#SemanticCapability
  bpmo#hasSemanticDescription hasValue
  "http://kmi.open.ac.uk/swsc/datamediator#PurchaseOrderRequest"
  instance SemanticCapabilityReceivePO_WSMO memberOf
  bpmo#SemanticCapability
  bpmo#hasSemanticDescription hasValue
  "http://kmi.open.ac.uk/swsc/wsmo/RequestPOWS#RequestPOWS"
  instance GoalTask_CreateOrder memberOf bpmo#GoalTask
  bpmo#hasName hasValue "Create Order"
  bpmo#hasHomeProcess hasValue Process_MoonMediator
  SemanticCapability_CreateOrder_WSMO
  bpmo#hasPartnerRole hasValue moonCRM
  bpmo#hasInputChange hasValue SemanticCapability_CreateOrderDesc
  SemanticCapability_OrderDesc
  instance SemanticCapability_OrderDesc memberOf
  bpmo#SemanticCapability
  bpmo#hasSemanticDescription hasValue
  "http://kmi.open.ac.uk/swsc/wsmo/RequestPOWS#RequestPOWS"

We use the instances in Listing 5 to illustrate dataflow as well as the use of a MediationTask to map instances of PurchaseOrderRequest used by the requester (Receive.ReceivePO) to Order, used by the provider (GoalTask_CreateOrder), as shown in Listing 6 and Listing 7. MediationTask_MapPurchaseOrder provides two DataMediators: DataMediator_MapOrderRequestToSearchCustomer and DataMediator_MapOrderRequestToOrder.

### Listing 6 BPMO instances related to data mediation

```
<instance MediationTask_MapPurchaseOrder memberOf bpmo#MediationTask
  bpmo#hasName hasValue "Map Purchase Order"
  bpmo#hasHomeProcess hasValue Process_MoonMediator
  bpmo#hasDataMediator hasValue
  [DataMediator_MapOrderRequestToSearchCustomer,
   DataMediator_MapOrderRequestToOrder]

We use the instances in Listing 5 to illustrate dataflow as well as the use of a MediationTask to map instances of PurchaseOrderRequest used by the requester (Receive.ReceivePO) to Order, used by the provider (GoalTask_CreateOrder), as shown in Listing 6 and Listing 7. MediationTask_MapPurchaseOrder provides two DataMediators: DataMediator_MapOrderRequestToSearchCustomer and DataMediator_MapOrderRequestToOrder.

```

```
This query asks about any instance of Order with corresponding attribute values. The result of the query is presented in Figure 2, which basically shows a newly inferred instance of Order (named using the function symbol moonOrder, in accordance with the consequent of the axiom) to which the attribute values of instance bluePORequest is mapped.

For illustration purposes, we performed the query below over the sample instances to test the OrderFromPurchaseOrderRequest axiom, using the IRIS reasoner:

```
axiom OrderFromPurchaseOrderRequest

impliesBy

?request hasValue ?r
and ?r hasValue Order

memberOf Order

implies

moonOrder(?request)

impliesBy

?auth ?name
and ?name hasValue ?auth

memberOf Order

implies

SendPONeededInformation

impliesBy

?auth ?name
and ?name hasValue ?auth

memberOf Order

implies

ReceivePONeededInformation
```

Figure 2 Query result using mappings

We next illustrate a query that can be performed over BPMO instances related to its component workflow activities. A business analyst might be interested in knowing about tasks and partners of a specific process. For example, in the query below we ask which tasks are related to partner role customer (company Blue), with the corresponding attributes values of hasName (?name) and hasPartnerWebService (?ws):

```
?task hasValue ?name
and ?name hasValue customer

implies

SendPONeededInformation

impliesBy

?auth ?name
and ?name hasValue ?auth

memberOf PartnerRoleDescription

implies

ReceivePONeededInformation
```

Figure 3 Query result for finding workflow activities

The result of this query (Figure 3) contains the instances of Receive (Receive Purchase Order) and Send (Send PO Confirmation) corresponding to the interaction with company Blue as expected.

Another interesting query, of which we omit the details only for brevity, would be a query over processes according to the associated organisational attributes, such as business policies and strategies. We note that, like semantic search, many implicit inference may be involved in such queries. Queries are not, therefore, limited in scope by the data that are explicitly represented, as they would be in a database. Furthermore, by representing all of these modelling concerns of BPM in a uniform and connected model we can issue cross-cutting queries that are

---

6 Such a rule is beyond the capabilities of a DL-based ontology language

7 IRIS is an open source reasoner for WSML Flight, available as an integrated component of WSMO Studio

---
impossible in a tool dedicated to, say, process modelling where other aspects are separately modelled and maintained.

5. CONCLUSIONS AND RELATED WORK
BPMO describes a rich business process model, as demanded by the BPM community, using ontological descriptions to capture workflow and organisational concerns in a uniform and extensible manner, and reuses the results of Semantic Web Services research for the description of interaction activities.

There are various advantages for using BPMO. First, BPMO provides comprehensive semantic annotations for business processes that can be used for automated inference at the business level while facilitating the translation to the execution level. Second, BPMO provides links from the process to organisational aspects, which can be modelled independently for different domains. Third, BPMO can be used to verify at the semantic level restrictions applied to the workflow or certain process activities. Finally, BPMO facilitates the modelling of new (or mediation) processes based on existing ones as well as the discovery of services for goal-based activities.

BPMO facilitates semantic interoperability by modelling interaction activities using SWS descriptions of inputs, outputs and operations. These activities use ontologically defined data for dataflow and also take advantage of the semantic mappings provided by the BPMO Data Mediators.

BPMO diagrams can be created using practical and freely available tooling with WSMO Studio. The advantage is that the BPMO modeller of WSMO Studio automatically generates BPMO instances from the workflow diagram and allows easy reference to ontology instances and service descriptions. BPMO uses WSML-Flight as the representation language, which can be used with the IRIS reasoner for performing instance validation and queries.

From the business viewpoint, business analysts can perform semantically enabled queries directly and uniformly on the business context and activities of a business process. The queries can be extended to BPMO’s translation destinations and sources throughout the process life cycle from creation to deployment, monitoring and execution. In this way, reuse across the business/IT divide is facilitated and great scalability is achieved through increased automation supported by ontology-based reasoning.

There is substantial work discussing the translation and mismatches between BPMN and BPEL (e.g. [12], [13]), and more generically between block and graph oriented workflow notations [8], that has informed the implementation of BPMO concepts and attributes, especially in what concerns enabling the translation of BPMO constructs from notations such as BPMN, and to languages such as BPEL. For instance, we have developed a number of translators (e.g. [9]) within the SUPER project that use appropriate workflow pattern representations in BPMO to avoid workflows with acyclic loops and unsynchronised branches. One main difference from existing standards to BPMO, though, is that we use ontologies and extensions to support Semantic Web Services.

OWL-S®, the SWS ontology submitted to W3C, contains a semantic-based process workflow description (i.e. the process model), which serves the same purpose as BPMO; however this model is not very rich. As pointed out in [3], there are a number of constructs from BPEL, such as conditions, synchronization and external (event-based) choices and handlers that cannot be expressed in OWL-S.

The Semantic Web approach presented in [3] has similar goals to our approach using BPMO in that the authors there argue that the syntactic approach provided by BPEL has shortcomings that limit its ability to provide seamless interoperability. They propose the use of semantic-based technologies (OWL-S) to support automated service discovery, customization and semantic translation for BPEL based processes; however, their annotations for services and data are decoupled from the control-flow language. BPMO, instead, provides semantically annotated control-flow constructs coupled with semantic descriptions of data and services. In addition, BPMO workflow includes semantic data mediation via Mediation Tasks and Data Mediators, which can refer to mapping rules and mediation services.

In [15] an approach called Semantic Process Templates (SPT) is presented, which provides semantic extensions to a (XML-based) BPEL-compatible process workflow specification. A semantic template is used for every activity in the process definition in order to attach concepts from a given ontology to inputs, outputs and operations. SPT differs from BPMO because it is a bottom-up approach, tied to a particular execution standard (BPEL), and the control-flow constructs have no ontological representation.

6. ACKNOWLEDGMENTS
The work presented in this paper was partly funded by the European Commission under the SUPER project (FP6-026850).

7. REFERENCES

http://www.w3.org/Submission/OWL-S/


