Translating Semantic Web Service Based Business Process Models

Liliana Cabral and John Domingue
Knowledge Media Institute
The Open University
Milton Keynes, UK
l.s.cabral@open.ac.uk and j.b.domingue@open.ac.uk

Abstract— We describe a model-driven translation approach between Semantic Web Service based business process models in the context of the SUPER project. In SUPER we provide a set of business process ontologies for enabling access to the business process space inside the organisation at the semantic level. One major task in this context is to handle the translations between the provided ontologies in order to navigate from different views at the business level to the IT view at the execution level. In this paper we present the results of our translation approach, which transforms instances of BPMO to instances of sBPEL.

Keywords: model translation; Semantic Web Services, ontologies; process models; ATL rules

I. INTRODUCTION

One of the concerns of Business Process Management (BPM) is to provide process modelling languages and tools to facilitate bridging between the business and Information Technology (IT) views. However, one major obstacle to the complete realization of BPM today is that the business process space inside the organisation, from the business expert perspective to the actual implementation is widely not accessible at the semantic level and thus neither to machine reasoning. The emerging Semantic Business Process Management (SBPM) research area [7] addresses this problem and proposes the use of ontologies and Semantic Web Services (SWS)[5] in order to provide a unified view on business processes in a machine understandable way.

Within the SUPER project[1], an approach to SBPM has been developed, which in particular provides a set of integrated ontologies developed in WSML[2] taking into account the use of Semantic Web Services for business process modelling. More specifically, we provide ontologies for a number of popular standards (e.g. BPMN, EPC, BPEL) as well as the novel Business Process Modelling Ontology (BPMO) [4], which provides a high-level model of business processes, integrating organisational aspects, process workflow and services. The goal is to support a number of BPM life-cycle activities at the semantic level, including modelling, querying, translation and execution. One major task within this approach is thus handling the translations between the provided ontologies in order to navigate from different views at the business level to the IT view at the execution level.

In this paper we describe a model-driven approach and implemented translator for transforming instances of BPMO to instances of an ontology for BPEL (sBPEL). In particular, our approach implements mappings using ATL[3] rules and uses the XML format of WSML as both source and target models of the ATL transformation engine. The result of the translation is a portable file containing a semantically annotated executable business process model.

The rest of the paper is structured as follows. Section 2 describes the context and a subset of the business process ontologies used in the SUPER project. Section 3 describes the translation approach we adopted and the implementation of the BPMO2SBPEL translator. Section 4 describes a translation example from a use case. Finally, we present our conclusions and related work.

II. BUSINESS PROCESS ONTOLOGIES

As mentioned in the introduction, within the SUPER project we provide a set of integrated ontologies, which represent different views and levels of business process models. In Figure 1 we depict a subset of the available ontologies (rounded rectangles) for the purpose of explaining our translation approach. The main ontology is BPMO (see Section 2.A) to and from which corresponding translations are performed (large arrows in the picture). BPMO imports UPO (Upper-level Process Ontology), an ontology defining common business process concepts, shared by all ontologies. sEPC [6] and sBPMN [2] are the two ontologies created to semantically annotate the corresponding standard notations used to model process workflows at the business level. SBPEL [10] (see Section 2.B) is an ontology for the BPEL language (with extensions), which is used by IT experts to execute process workflows. These ontologies can be grounded to any tool-specific syntactic format of the respective notation (rectangles in the picture) via straightforward serializations.

This paper will focus on the translation between BPMO and sBPEL, taking advantage of the unambiguous meaning of

[1] Semantics Utilised for Process Management within and between Enterprises (http://www.ip-super.org)
[2] http://www.w3.org/Submission/WSML/
[3] http://www.eclipse.org/M2M/ATL

constructs in the ontologies, in order to facilitate the navigation from the business level to the execution level.

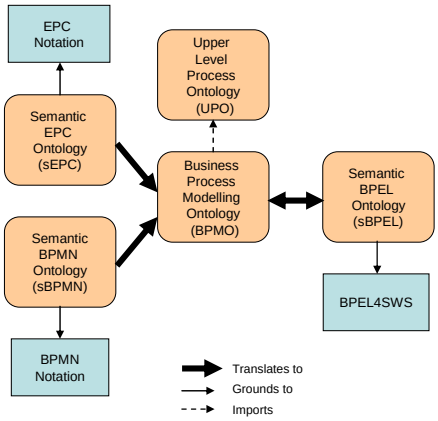

Figure 1. Business Process Ontologies in SUPER

BPMO and the related ontologies mentioned above are publicly available at the SUPER website (http://www.ip-super.org/ontologies). Next, we describe relevant details of BPMO and sBPEL, which are the ontologies we use in this paper.

A. BPMO

BPMO[4] is an ontology 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 [15] and are informed by, and named according to Workflow Patterns [1]. Moreover, BPMO concepts related to interaction activities (tasks) adopt a number of Web Service attributes also used in BPEL constructs.

Basically, a BPMO process description captures the business/organisational 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). BPMO process workflow elements are structured into a workflow container combining features of block-oriented and graph-oriented workflow patterns. 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 *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. 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* following the first element.

Listing 1. BPMO Process and Business Activity Concepts

```
concept BusinessActivity subConceptOf upo#BusinessActivity
  hasName ofType  (0 1) _string
  hasDescription ofType  (0 1) _string
  hasNonFunctionalProperties ofType(0 1)
BusinessActivityNonFunctionalProperties
  hasBusinessDomain ofType upo#BusinessDomain
  hasBusinessFunction ofType upo#BusinessFunction
  hasBusinessStrategy ofType upo#BusinessStrategy
  hasBusinessPolicy ofType upo#BusinessPolicy
  hasBusinessProcessMetrics ofType
upo#BusinessProcessMetrics
     hasBusinessProcessGoal ofType upo#BusinessProcessGoal
  hasBusinessResource ofType upo#Resource

concept Process subConceptOf {BusinessActivity, upo#
BusinessProcessModel}
  hasWSDescription ofType(0 1) SemanticCapability
  hasWorkflow ofType  (0 1) Workflow

concept Workflow subConceptOf
upo#ProcessOrchestrationSpecification
  hasHomeProcess ofType  (0 1) Process
  hasFirstWorkflowElement ofType(1 1) WorkflowElement
```

The concepts related to Semantic Web Services in BPMO are *GoalTask*, *Receive*, *Send* and *ReceiveMessage Event* (see Listing 2), which are subconcepts of *Task*. A *Task* is also a *Business Activity* (as in Listing 1). *Tasks* have attributes to represent information about the interaction with a partner process, such as partner role (*hasPartnerRole*), inputs (*hasInputDescription*) and outputs (*hasOutput Description*). Most attribute types in *Tasks* are defined as *SemanticCapability* which is a wrapper for abstracting over domain data instances or service descriptions.

Listing 2. BPMO Concepts Related to Interaction Tasks

```
concept BusinessRole subConceptOf upo#Role
  hasName ofType  (0 1) _string
  hasDescription ofType  (0 1) _string
  hasOrganisation ofType (0 1) upo#Organisation

concept GoalTask subConceptOf Task
  hasPartnerGoal ofType  (0 1) SemanticCapability
  hasPartnerRole ofType (0 1) BusinessRole
  messageTo ofType  (0 1) Receive
  messageFrom ofType  (0 1) Send
  hasInputDescription ofType SemanticCapability
  hasOutputDescription ofType SemanticCapability
  requestsCapability ofType  (0 1) SemanticCapability
  providesCapability ofType  (0 1) SemanticCapability

concept Send subConceptOf Task
  hasPartnerWebService ofType (0 1)SemanticCapability
  hasPartnerRole ofType (0 1) BusinessRole
  hasReceiveCounterpart ofType (0 1) Receive
  messageTo ofType  (0 1) Receive
  hasOutputDescription ofType SemanticCapability
  requestsCapability ofType  (0 1) SemanticCapability

concept Receive subConceptOf Task
  hasPartnerWebService ofType (0 1 SemanticCapability
  hasPartnerRole ofType (0 1) BusinessRole
  hasSendCounterpart ofType Send
  messageFrom ofType  (0 1) Send
  hasInputDescription ofType SemanticCapability
  providesCapability ofType  (0 1) SemanticCapability

concept ReceiveMessageEvent subConceptOf {IntermediateEvent,
Receive}
```

[4] http://ip-super.org/ontologies/process/bpmo/v2.0.1#bpmo

A *GoalTask* is 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 a Goal (or request) description. The *hasInputDescription* and *hasOutputDescription* attributes refer to the semantic descriptions of request and response data respectively. The *requestsCapability* and *providesCapability* attributes refer to the semantic descriptions of Web Service 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.

B. SBPEL

Semantic BPEL (sBPEL)[5] is an ontology for BPEL4SWS [11], which is an extension of BPEL4WS[6], a language for specifying the composition of Web Services using a control-flow based approach. Control-flow constructs are either structured activities (e.g. sequence, *flow*, *if*), or basic activities including assign and interaction activities (e.g. *receive*, *reply*, *invoke*, *pick*). In addition, *links* (dependencies) can be added between activities within a *flow* (parallel execution) activity in a graph-based style. BPEL4SWS uses the extensibility elements of BPEL in order to add semantic annotations of data and services to the process. It also adds the *Interaction* and *Conversation* extension constructs, which can be used to group a number of interaction activities for modelling long running conversational interaction among partners.

Listing 3 sBPEL Concepts related to a Process

```
concept SemanticProcess subConceptOf bpel#Process
  hasConversation ofType Conversation
  hasPartner ofType Partner
  hasSemanticOnMessage ofType SemanticOnMessage

concept Receive subConceptOf {bpel#Interaction,
bpel#NewActivityType}
  doesCreateInstance ofType  (0 1) _boolean
  belongsToConversation ofType (1) Conversation
  hasVariable ofType (1) SemanticVariable

concept Conversation
  hasName ofType (1) _string
  describesInterface ofType (1) InterfaceDescription
  correspondsTo ofType bpmo#Process

concept IncomingInterface subConceptOf InterfaceDescription
  hasWebServiceDescription ofType (1) _string

concept SemanticVariable subConceptOf bpel#Variable
  hasSemanticType ofType (1) _string

concept Partner
  hasName ofType  (1) _string
  hasBusinessEntity ofType (0 1) _string
  hasConversation ofType  (1 *) Conversation

concept ExtensionActivity subConceptOf bpel#BasicActivity
  hasActivity ofType (1) NewActivityType
```

For example, as shown in Listing 3, the *Receive* concept (sub-concept of *Interaction* and *NewActivityType*) ontologically represents one of the interaction activities of BPEL4SWS in sBPEL. The *Receive* activity can be added to the control-flow via the *ExtensionActivity* concept (*hasActivity* attribute). *Receive* must belong to a *Conversation* (*belongsToConversation* attribute), which in turn must define an interface (*describesInterface* attribute) that has a Semantic Web Service description (*hasWebServiceDescription* attribute). Note also that *Receive* (*hasVariable* attribute) uses *SemanticVariable* (*hasSemanticType* attribute) to annotate the data received. The example of an instance is given in Section 4.

III. TRANSLATION APPROACH

In this section we describe the implementation of the BPMO2SBPEL translator, which transforms between instances of a BPMO model and a sBPEL model. The translator takes as input a WSML file containing BPMO instances of an individual business process and then generates as output a WSML file containing corresponding sBPEL instances (see example in Section 4). The generated sBPEL file can be used in a later stage for serialization and then execution in an appropriate engine.

We have developed a standalone java API for BPMO2SBPEL (available at http://kmi.open.ac.uk/projects/super/BPMO2SBPEL-2.0.zip). This API uses the WSMO4J API (http://wsmo4j.sourceforge.net) to parse and serialize XML versions of WSML as required; creates the input files from the XML files; and launches the ATL engine.

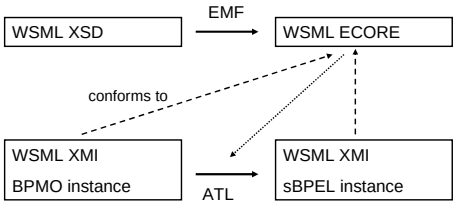

Figure 2. ATL Transformation of WSML instances

The translator has been developed using EMF[7] (Eclipse Modelling Framework). The translator rules are written in ATL[8] (Atlas Transformation Language), which is a hybrid language (a mix of declarative and imperative constructors), designed to express model transformations. As illustrated in the diagram of Figure 2, the ATL engine requires meta-models in the ECORE format and conforming XMI source and target models for the transformation. Since our source and target models are in WSML, we create a WSML meta-model using a specific EMF tool, which takes a XSD file. We use the XML syntax of WSML[9] in order to generate the meta-model (Wsml.ecore) from WSML XSD. In addition, the input WSML XMI instances are generated (programmatically) from given WSML XML instances. There is a small issue with mixed XSD types within WSML XSD, which cannot be processed by ATL. The work around this problem was to create another version of the WSML XSD and corresponding meta-model

[5] http://ip-super.org/ontologies/ process/sbpel/v2.0.0#sbpel
[6] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
[7] http://www.eclipse.org/modelling/emf
[8] http://www.eclipse.org/M2M/ATL
[9] http://www.wsmo.org/TR/d16/d16.1/v0.21/xml-syntax/wsml-xml-syntax.xsd

(syntax.ecore) using strings for primitive types instead of mixed ones, and use both ECORE source models for the translation.

The complete specification of the ATL rules that provides the mappings for our translator is available within the distribution package together with the API mentioned previously. The reader is referred to the Eclipse website mentioned before for details of the ATL language.

A translation from BPMO to sBPEL can be based on structured or graph elements. Our implementation for translating a structured BPMO process is restricted to structured and pattern-based elements of BPMO. That is, we consider BPMO diagrams (instances) which contain Tasks (*GoalTask*, *Receive*, *Send*, *MediationTask*), *Events* and block patterns between a *StartEvent* and an *EndEvent* or within an initial *Sequence*. The block patterns supported are: *Sequence*, *ParallelSplit Synchronise*, *MultipleChoiceMerge*, *DeferredChoiceMerge*, *Exclusive*, *ChoiceMerge*, *Repeat* and *While*. In addition, the branches can only contain recursive single block-patterns. In this case, the instances of BPMO and the translated instances of sBPEL are structured. Note that BPMO processes that need not be translated (executed), have no such restrictions. On the other hand, a graph-based translation would be from a BPMO diagram, which would contain elements similar to the above list, with no restriction for composed branches using block and graph patterns. In this case, the BPMO instance would contain elements explicitly linked using control-flow *Connectors*. One possible implementation for this case is to detect the composed branches in BPMO and translate them into sequences in sBPEL.

The translation involves a number of mapping cases, expressed in the ATL rules: sBPEL *Partner*, *Role* and *Conversation* concepts are derived from diverse attributes in BPMO Tasks; BPMO *Tasks* can derive multiple chained target concepts in sBPEL; ordered elements in BPMO (from *sequences* and *conditional branches*) generate linked-lists in sBPEL; and some target concepts in sBPEL must point back to the source concept in BPMO. Otherwise, both ontologies are aligned in the capacity of handling ontological data, SWS descriptions and semantic based mappings.

IV. TRANSLATION EXAMPLE

In this section we present an example of a simple business process model in order to illustrate the translation from BPMO to sBPEL. Figure 3 depicts the workflow diagram of a business process taken from a use case in the telecommunication domain within SUPER. The *Content Provision Process* is the model of a Telco service provider for downloading Web content for a customer. This process diagram was created using WSMO Studio's BPMO Modeller[10], which generates an initial set of BPMO instances corresponding to the process control-flow, to which the user can add attribute values (links to ontology instances and SWS descriptions) using the modeller's property editor.

[10] http://www.wsmostudio.org/

According to the process workflow the following tasks take place: a request is received (*Receive*); the input is mediated (mapped to inputs of next tasks) (*MediationTask*); two invocations (*GoalTask*s) in parallel (*ParallelSplitSynchronise*) are performed to get the license and URL of the content; outputs are mediated (aggregated); and finally the result is sent (*Send*) to the customer. The process (service provider) is in fact interacting with three partners: the content requester (customer), the license provider and the content provider.

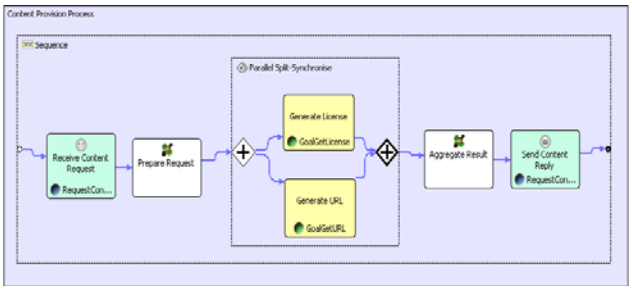

Figure 3. Example of a BPMO Process Diagram

The BPMO instance representing the first task (Receive Content Request) is shown in Listing 4, and the result of the corresponding translation is shown in Listing 5. First, the instance of *Process* in BPMO is translated to an instance of *SemanticProcess* in sBPEL. A *SemanticProcess* has more details than the corresponding BPMO, so attributes like *hasPartner* and *hasConversation* are generated from information from other elements such as *Receive* in BPMO. The structured *Sequence* concept in BPMO is translated to the *Sequence* concept in sBPEL, implemented as a linked list. The *ParallelSplitSynchronise* in BPMO is translated to *Flow* in sBPEL. The translation of *Receive* generates a chain of concepts in sBPEL, which are *ExtensionActivity*, *Receive*, *Conversation* and *IncomingInterface*. These correspond respectively to the translation of *Receive_ContentRequest* (BPMO) to the instances *Receive_ContentRequest_ sBPEL*, *Receive_ContentRequest_sBPELReceive*, *Receive_ContentRequest_sBPELConversation* and *Receive_ContentRequest_sBPELInterface*. Similar chains of concepts are created for the translation of *GoalTask* and *Send*. Note also the use of the *correspondsTo* attribute to point back to the originating BPMO instance.

Listing 4. Example of a BPMO instances

```
instance Receive_ContentRequest memberOf bpmo#Receive
  bpmo#hasName hasValue "Receive Content Request"
  bpmo#hasHomeProcess hasValue Process_ContentProvision
  bpmo#hasPartnerWebService hasValue
SemanticCapability_ContentRequester_WSMO
  bpmo#hasInputDescription hasValue
SemanticCapability_ContentRequestMessage
  bpmo#providesCapability hasValue
SemanticCapability_ContentRequestOperation
  bpmo#hasPartnerRole hasValue contentRequester

instance SemanticCapability_ContentRequestMessage memberOf
bpmo#SemanticCapability
  bpmo#hasSemanticDescription hasValue "http://ip-
super.org/kmi/ContentProvision/RequestContentWS#contentReque
stMessage"
instance contentRequester memberOf bpmo#BusinessRole
  bpmo#hasName hasValue "Content Requester"
  bpmo#hasOrganisation hasValue kmi
```

The concept *SemanticCapability* in BPMO translates to *SemanticVariable* in sBPEL. For example, the value of the attribute *hasInputDescription*, which provides the URL of the concept that describes the input message, is translated to *SemanticCapability_ContentRequestMessage_sBPEL* (*hasVariable* attribute) in sBPEL. This *SemanticVariable* (*hasSemanticType* attribute) has the translated value.

Listing 5. Example of sBPEL instance generated by the translator

```
instance Receive_ContentRequest_sBPEL memberOf
bpel#ExtensionActivity
  bpel#correspondsTo hasValue _"http://ip-
super.org/examples/process/bpmo/v2.0.1/examples#Receive_Cont
entRequest"
  bpel#hasActivity hasValue
Receive_ContentRequest_sBPELReceive

instance Receive_ContentRequest_sBPELReceive memberOf
sbpel#Receive
  sbpel#hasName hasValue "Receive Content Request"
  sbpel#hasVariable hasValue
SemanticCapability_ContentRequestMessage_sBPEL
  sbpel#belongsToConversation hasValue
Receive_ContentRequest_sBPELConversation

instance SemanticCapability_ContentRequestMessage_sBPEL
memberOf sbpel#SemanticVariable
  bpel#hasName hasValue
"SemanticCapability_ContentRequestMessage _sBPEL"
  bpel#hasType hasValue
SemanticCapability_ContentRequestMessage
_sBPELWSDLMessageType
  sbpel#hasSemanticType hasValue "http://ip-
super.org/kmi/ContentProvision/RequestContentWS#contentReque
stMessage"

instance SemanticCapability_ContentRequestMessage_
sBPELWSDLMessageType memberOf bpel#WSDLMessageType
  bpel#hasDefinition hasValue "#wsdl11.message()"

instance Receive_ContentRequest_sBPELConversation memberOf
sbpel#Conversation
  sbpel#hasName hasValue
"Receive_ContentRequest_sBPELConversation"
  sbpel#describesInterface hasValue
Receive_ContentRequest_sBPELInterface

instance Receive_ContentRequest_sBPELInterface memberOf
sbpel#IncomingInterface
  sbpel#hasWebServiceDescription hasValue "http://ip-
super.org/sws/ContentProvision/wsmo/RequestContentWS#Request
ContentWS"
```

BPMO as a model which represents business processes at the business level supports constructs that might not apply to sBPEL. Thus, similar to the mappings between BPMN and BPEL [12], the mappings from BPMO to sBPEL can only be partial. For example, BPMO allows business analysts to create arbitrary cycles, which are not supported in sBPEL. The way to restrict the translation is to use only translatable constructs such as block patterns in BPMO or doing a pre-validation through the use of axioms, which can guarantee that a valid BPMO instance contains only suitable elements for the translation. These axioms can be contained in a separate ontology and imported for validation of the BPMO instance. This is based on the fact (see for example [9]) that there is no generic translation from a graph-based notation such as allowed in BPMO to one such as sBPEL, which provides mostly structured activities to model a process and some restricted use of links to enable a graph-based style.

V. Conclusions

In this paper we show that we can effectively translate a business-level process model annotated with BPMO to an executable process model annotated with sBPEL, using a model-driven approach based on ATL rules. We presented the two ontologies and discussed elements related to Semantic Web Services. The preliminary implementation of our translator is limited to structured, pattern-based elements of BPMO. We illustrated our work using an example from a use case, but the ontologies (http://www.ip-super.org) as well as the translator's API and ATL rules are freely available in a distribution package with complete examples at http://kmi.open.ac.uk/projects/super/BPMO2SBPEL-2.0.zip.

Regarding the use of ATL rules, we recognize that we do not follow on the original prescribed use of models in ATL, since in our approach the two used models (BPMO and SBPEL) use the same metamodel (WSML). Thus, we get low type safety from the models, but compensate by relying on the semantics given by the ontologies. Yet, our approach has proven to be quite effective as ATL provides not only a rule engine but also constructs that allows us to replace names (including namespaces) and check model hierarchies. In previous work we provided a translation based on WSML rules, but there the mappings could only imply (sBPEL) instances, which would have to be consumed at runtime inside a WSML based environment. With the approach in this paper we are able to generate a new ontology (file) with mapped instances, which can be validated against the SBPEL ontology and later extended.

VI. Related Work

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 [9], which have informed the implementation of our ontologies and BPMO2sBPEL translator. For instance, the translation exploits appropriate workflow pattern representations in BPMO to avoid workflows with acyclic loops and unsynchronised branches. One main difference from that work to ours, though, is that we use ontologies and extensions to support Semantic Web Services.

The Semantic Web approach presented in [8] has similar goals to our approach using BPMO, as the authors there argue that the syntactic approach provided by BPEL4WS 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 BPEL4WS based processes; however, their annotations for services and data are decoupled from the syntactic control-flow language. BPMO, instead, provides semantically annotated workflow activities coupled with semantic descriptions of data and services. In addition, BPMO allows for semantic data transformation by the use of *Mediation Tasks* with *Data Mediators* that can be translated to extended *assign operations* in sBPEL.

The translation approach presented in [3] provides mappings from BPEL4WS to OWL-S, aiming at providing semantics to business process models. This work presents a bottom-up translation, from the syntactic level to the semantic level; however, it shows that there are a number of constructs from BPEL, such as synchronization, external (event-based)

choices and handlers, which cannot be mapped to OWL-S. In addition, input and output parameters in the resulting ontology need to be annotated with domain ontologies by the user.

ACKNOWLEDGMENT

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

REFERENCES

[1] W. Van der Aalst, A. Hofstede, B. Kiepuszewski and A. Barros, Workflow Patterns. Distributed and Parallel Databases, 14(3):5–51, 2003.

[2] W. Abramowicz, A. Filipowska, M. Kaczmarek and T. Kaczmarek, "Semantically enhanced Business Process Modelling Notation", in Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM-2007), Vol-251, CEUR-WS, June 2007.

[3] M. Aslam, S. Auer, J. Shen and M. Herrmman, "Expressing Business Process Models as OWL-S Ontologies", in Workshop on Grid and Peer-to-Peer Based Workflows (GPWW) in conjunction with BPM 2006, LNCS 4103, pp. 400-415.

[4] L. Cabral, B. Norton, and J. Domingue, "The Business Process Modelling Ontology", in Proceedings of the 4th International Workshop on Semantic Business Process Management (SBPM 2009) in conjunction with ESWC 2009, ACM Proceedings, 2009.

[5] D. Fensel, H. Lausen, A. Polleres, J. Bruijn, M. Stollberg, D. Roman, and J. Domingue, Enabling Semantic Web Services: the Web Service Modelling Ontology (WSMO), Springer, 2006.

[6] A. Filipowska, M. Kaczmarek and S. Stein, "Semantically annotated EPC within Semantic Business Process Management", in Workshop on Advances in Semantics for Web Services (semantic4ws), Milan, Italy, 2008.

[7] M. Hepp, F. Leymann, J. Domingue, A. Wahler and D. Fensel, "Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management", in Proceedings of IEEE Intl. Conf. on e-Business Engineering (ICEBE 2005) pp. 535-540.

[8] D. Mandell and S. McIlraith. "Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation", in proceedings of the 2nd International Semantic Web Conference (ISWC 2003), LNCS 2870, Springer.

[9] J. Mendling, K. Lassen and U. Zdun, "On the transformation of control flow between block-oriented and graph-oriented process modelling languages", in International Journal of Business Process Integration and Management, Vol. 3(2), 2008.

[10] J. Nitzsche, D. Wutke and T. Lessen, "An ontology for executable business processes", in Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM-2007), Vol-251, CEUR-WS, June 2007.

[11] J. Nitzsche, T. Lessen, D. Karastoyanova and F. Leymann, "BPEL for Semantic Web Services - BPEL4SWS", in Proceeding of OTM Workshops (OTM 2007), Part I, LNCS 4805, 2007.

[12] C. Ouyang, W. Van der Aalst, M. Dumas and A. Hofstede, "From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way", available at http://eprints.qut.edu.au/archive/00005266/01/5266.pdf (2006)

[13] J. Recker and J. Mendling, "On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages", available at http://eprints.qut.edu.au/archive/00004637/01/4637.pdf (2006).

[14] A. Scheer, T. Oliver and A. Otmar, Process Modelling Using Event-Driven Process Chains, (chapter) in Process-Aware Information Systems, Wiley, 2005.

[15] P. Wohed, W.Van der Aalst, M. Dumas, M. Hofstede and N. Russell, "On the suitability of BPMN for Business Process Modelling", In Proceedings of the Fourth Business Process Management Conference (BPM 2006), LNCS 4102, pp.161-176, Springer.