Mapping Web Services Specifications to Process Ontology: Opportunities and Limitations

Jun Shen  Yun Yang  Bharat Lalwani
CICEC – Center for Internet Computing and E-Commerce
School of Information Technology
Swinburne University of Technology, PO Box 218, Melbourne, Australia, 3122
{jshen, yyang}@it.swin.edu.au, amitlalwani81@hotmail.com

Abstract

There are many XML based workflow and process modeling languages like XPDL, BPEL4WS etc. that can be used to define workflow and business processes in the Web services world. However, the use of XML makes the meaning of the processes ambiguous with limited capability to describe the relationships (semantics, schemas or ontologies) with respect to objects. Put into action, it becomes obvious that the integration of Web services will be hardly achieved without certain schema matching mechanisms supported by higher level abstraction. In this paper, the authors have developed a tool to support the mapping of business processes defined in BPEL4WS onto DAML-S based Process Ontology. This will overcome the weaknesses of the evolving BPEL4WS language; provide a powerful way to describe the objects and their relationships; and reduce the overhead of combinations required when mapping a BPEL4WS specification to other process definition languages.

1. Introduction

The Web services paradigm is poised to become the dominant form of distributed computing this decade and beyond. A. T. Kearney, an EDS global consultancy, found that 75% of companies ranging from less than $50 million to more than $1 billion in revenues and across 20 vertical industries have already deployed one or more Web services [15]. Web services are helping to bridge the gap between business people and technologists in an organisation. They also make it easier for business people to understand technical operations [17]. Web services involve a family of related protocols to describe, deliver, and interact with services. The most well-known are Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). WSDL is most important in our context. WSDL files include a set of standard elements. These elements describe a particular Web service, including its interfaces and usage [5].

Workflow Management Systems have become the most promising solutions for the organisations that need to automate their business processes [18]. Applying workflow to a business process brings the details of that process into focus and adds the required business rules and logic in the process [3]. However, today's Web services do not allow defining the business process semantics of Web services, thus, they are isolated and opaque. Breaking isolation means connecting Web services and specifying how collections of Web services are jointly used to realise more complex functionality - typically a business process [2].

Nowadays, there are many XML based workflow process definition languages (process modeling languages) like BPEL4WS (Business Process Execution Language for Web Services) [1], XPDL (XML Process Description Language) [8], ebXML, etc. that can be used to describe workflow systems and business processes in the Web services world. They are specialised languages for describing all aspects of the workflow and representing the business process logic, defining the sequence of tasks and the routing rules that must be followed, as well as other business rules implemented by the workflow system. The Meta models of each language vary significantly from one specification to another and are all similar to the concept of Web services. The integration of Web services into higher levels of abstractions or frameworks will not be achieved without certain schema matching mechanisms and will lead to interoperability issues that should be considered for effective data sharing and exchange between the workflow systems [4]. It will require a lot of mappings to be done if we have many workflow process modeling languages in the domain.

Fortunately, the Defense Advanced Research Projects Agency (DARPA) Agent Markup Language for Services (DAML-S) developed by universities and IT companies, is an attempt to provide an ontology for describing Web services [6, 7]. DAML-S has a well-defined semantics, making it computer interpretable and unambiguous.

The authors have developed a tool to support the mapping of business processes defined in BPEL4WS onto
DAML-S based Process Ontology, which is a containment framework to describe Web services in form of workflows. This will overcome the known weaknesses and the limitations of BPEL4WS and ontology may powerfully describe the objects and their relationships and reduce the overhead of mapping combinations required.

This paper is organised as follows. Section 2 discusses the typical Web service specification language BPEL4WS, process ontology DAML-S and their relationships briefly. Section 3 details the mapping specification from BPEL4WS and DAML-S. Findings and limitations are also summarised there. Section 4 introduces our tool which supports the mapping. Section 5 discusses related work while section 6 concludes the paper.

2. Process definition and ontology

2.1. BPEL4WS

The BPEL4WS (v1.1) is a language for Web service orchestration released by major IT companies. It provides means to model the behaviour of Web services in a business process interaction. It allows businesses to describe sophisticated business processes that can both consume and provide Web services. The language is intended to support the modeling of both executable and abstract processes [1, 8]. An abstract process is a business protocol that specifies the message exchange behaviour between different parties without revealing their internal behaviour. An executable process specifies the execution order between a number of activities that constitute the process, the partners involved in the process, the messages exchanged between these partners, and the fault and exception handling that specify the behaviour to adopt in cases of errors and exceptions [1].

A BPEL4WS process is a kind of flow-chart, where each element in the process is called an activity. An activity can be either primitive or structured. A BPEL4WS process model is layered on top of the service model defined by WSDL 1.1. A BPEL4WS process definition provides and/or uses one or more WSDL services, and provides the description of the behaviour and interactions of a process instance relative to its partners and resources through Web service interfaces. BPEL4WS leverages WSDL in three ways: (1) every BPEL4WS process is exposed as a Web service using WSDL that describes the entry and the exit points for the process; (2) WSDL data types are used to describe the information being passed within the process; and (3) WSDL might be used to reference external services required by the process.

BPEL4WS supports the implementation of any kind of business process in a very natural manner and has gradually become the basis of a standard for Web service description and composition. However, it has several shortcomings that limit the ability to provide a foundation for seamless interoperability. The semantics of BPEL4WS is not always clear, thus complicating the adoption of the language. Major limitations in BPEL4WS specifications have been listed in [10, 11]. At the heart of the problem is BPEL4WS’s reliance on describing services using XML and XML Schema. XML provides a rudimentary content language, but lacks the constructs to describe complex relationships between Web resources. While XML Schema augments XML with a data model and enables data typing, the semantics of XML is underspecified [13].

2.2. Process ontology

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. The use of ontologies provides a very powerful way to describe the objects and their relationships to the other objects. Using ontology in this effort will not only overcome the XML limitations but also solves the mapping problem and will propose a solution to BPEL4WS limitations and related problems. Having an ontology in the domain will reduce the number of combinations in the mapping, as we have to only map the business processes defined in any of the languages to the ontology which can be then mapped back easily to the other languages with the ability of ontology to provide shared set of terms describing the application domain with a common understanding for sharing information and knowledge within the workflow systems. An ontology oriented Web service modeling framework had been proposed in [9]. DARPA, in conjunction with the W3C, was developing DARPA Agent Markup Language (DAML) by extending RDF (Resource Description Framework) with more expressive constructs aimed at facilitating agent interaction on the Web [7]. These languages are the milestones in the deployment of ontologies [16].

DAML-S is an attempt to provide an ontology for Web services, within the framework of DAML. The ontology of services can be divided into three parts, which are characterised by the kind of knowledge provided about a service [6]. The Service Profile describes what the service requires of users or agents and what it provides to them. The Service Model describes the service's process model (the control flow and data flow involved in using the service). DAML-S Process Model defines three types of processes (atomic, simple and composite). It is designed to enable automated composition and execution of services and is related most closely to the BPEL4WS Process Model. Service Grounding connects the process model description to communication level protocols and message descriptions in WSDL. These components are annotated.
with classes of well-defined types that make the service descriptions machine-readable and unambiguous. Additionally, the ontological structure of types allows type definitions to draw properties from hierarchical inheritance and relationships to other types.

We will concentrate on DAML-S Service Model (also known as Process Model - process ontology). The top level class of DAML-S process ontology is process. A process can have any number of inputs and outputs, preconditions, effects and participants. There are three disjoint subclasses of the process class: (1) Atomic Processes which can be directly invoked, have no subprocesses and execute in a single step (from the perspective of the service requester); (2) Composite Processes which can be decomposed into other (atomic or composite) processes, which are linked by control constructs such as sequence or if-then-else (in contrast to atomic processes, they cannot be directly invoked); and (3) Simple Processes which cannot be directly invoked, but, like atomic processes, they are viewed as having single-step executions.

The features that distinguish and differentiate BPEL4WS from DAML-S in terms of expressiveness, semantics, automated composition and execution, fault handling and querying mechanisms can be found in [11]. The mapping of necessary WSDL elements to DAML-S process model will be used and extended in our mapping from BPEL4WS to DAML-S. Part of this work has been implemented in Carnegie Melon University [14].

3. The Mapping Specification

3.1. Overview

The following sub-sections will describe the detailed mapping of BPEL4WS elements and semantics to DAML-S process ontology and will also address the findings and limitations in this mapping process. Figure 1 illustrates the inputs and outputs for/of this mapping process.

![Figure 1. Inputs/Outputs of mapping process](Image)

BPEL and WSDL files defining the business processes and interfaces for each service respectively will be used as inputs in this process. Process DAML, illustrating the usage of DAML-S process ontology and describing the services defined in BPEL4WS together with Data Flow showing the DAML-S process annotations to relate various process parameters to each other will be produced as outputs of this mapping process (as shown in Figure 1).

3.2. Processes and Business Partners

Table 1 shows the mapping of the processes between BPEL4WS and DAML-S, followed by the criteria for this process mapping and how this mapping is identified.

<table>
<thead>
<tr>
<th>Table 1. Mapping of processes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BPEL4WS</strong></td>
</tr>
<tr>
<td>Executable Process</td>
</tr>
<tr>
<td>Abstract Process</td>
</tr>
</tbody>
</table>

A BPEL4WS executable process does not represent any abstract view of the process and can be directly invoked and hence will be mapped onto a DAML-S atomic or composite process, depending on the internal activity of the executable process. DAML-S atomic processes can be directly invoked and composite processes can be made explicitly invocable by setting the invocable property of the composite process to true. A BPEL4WS process can have only one main activity in it. Thus, the selection of the DAML-S process (atomic or composite) for the mapping will depend on this activity. On the other hand, BPEL4WS abstract processes cannot be directly invoked and also represents the abstract view of the process and hence will be mapped onto DAML-S simple processes. A DAML-S simple process can be thought of as a view on either an atomic or composite process [6]. Simple processes provide a means of characterising other processes at varying levels of granularity, for purpose of planning and reasoning. They give additional characterisation of how they work, in terms of other processes (atomic or composite) and are not directly invocable (abstract process).

A simple process that abstracts an atomic process is realised by that process which is done by using the realizedBy property of the simple process and the realizes property of the underlying atomic process. A simple process that abstracts a composite process is expanded to that process which is done by using the expandTo property of the simple process and the collapsesTo property of the underlying composite process [6]. Using the top level attribute – abstractProcess of the process definition, we can identify whether a BPEL4WS process is an abstract or executable process, thus, allowing us to map the process onto an appropriate DAML-S process.

Business Partners definitions are optional and need not cover all partner relationships/links defined using partners and partnerLinks elements in the process definition. A participant of a DAML-S process can be any kind of DAML object (Thing). However, the participant property of the DAML-S process can be specialised by specific DAML-S processes such as agents, objects, entities, etc [6].
3.3. Data Types and Variables

In WSDL to DAML-S [14], only mapping of port types and operations to their corresponding DAML-S atomics processes is presented. It does not reflect the mapping of WSDL messages in DAML-S process ontology. We have extended the mapping of WSDL to DAML-S, by including the WSDL messages in our DAML-S based workflow process ontology.

WSDL messages are used to represent the abstract definition of the data being transmitted in and out of the processes [5]. WSDL messages consist of one or more logical parts which are associated with a type from some type system (data type defined or built-in XML Schema: XSD). Since we know the data types of parts of the messages by the type attribute of the part element, all the messages will be represented as DAML class and type of the class will not be restricted to any particular data type other than DAML object (i.e. Thing). Parts of each message will be mapped onto the properties of their corresponding DAML classes and the data type of the property will be based on the type specified for that part. We can have custom and built data types defined in the XML Schema Declaration used by WSDL messages and we need to import these schema declarations in our process ontology. This will be done by using the namespaces declared for the schema declarations in WSDL file for its internal use and then importing them in our output Process DAML file.

Variables in BPEL4WS provide the means for holding messages that constitute the state of a business process. The messages held are often to be sent to the partners via primitive activities. The type of each variable may be a WSDL message type, an XML Schema simple type or an XML Schema element. The messageType, type or element attributes are used to specify the type of a variable. Variables defined by the messageType attribute (representing a WSDL message type) in BPEL4WS process will be mapped in the same way as the WSDL messages are mapped to the data types in DAML-S process ontology using the DAML class. We will avoid duplicate declarations by mapping the variables to the DAML class and then making the class same as the WSDL message (data type already declared) it represents; using the sameClassAs element of DAML. The data types of variables that are defined by using the type or element attribute will be a DAML object (i.e. Thing) by default.

3.4. Primitive Activities

BPEL4WS primitive activities are simple activities with single step executions. These activities having no sub-activities in it are commonly used as request-response operations, for assignments and for throwing exceptions in the process [1]. DAML-S atomic processes share the same concept, as they are the basic units of implementation, a black-box representation, which does not describe how such processes work. Receive, Reply and Invoke activities of BPEL4WS uses the portTypes and operations defined in the WSDL document to send and receive messages and invoke operations in the process.

The <receive> activity allows the business process to do a blocking and wait for a matching message to arrive. Assumed that the portTypes and operations from the related WSDL are mapped onto the corresponding DAML-S atomic processes; the <receive> activity will be mapped onto an atomic process which will be the sub-class of an atomic process derived from the WSDL. The super class atomic process will be identified by using the portType and the operation attributes of the <receive> activity. This will allow us to map the activity onto the accurate atomic process and sub-type process. A received message will be mapped as an input of the atomic process derived from the <receive> activity with no outputs for the process. The type of this input will be based on the variable attribute of the receive activity which specifies the name of the variable defined in the BPEL4WS process. As we use variables defined in the BPEL4WS process as data types in Process DAML, we can also identify their data type (in Process DAML). The mapping of the <invoke> activity combines the mapping of both <receive> and <reply> activities as discussed above. The inputVariable and outputVariable attributes will be used to determine the inputs and outputs and their types for the derived atomic process and will be the same as the inputs and outputs for the super class atomic process respectively (explained in sub-section 3.6).

The <throw> activity generates a fault from within the business process [1]. The type of the fault that is generated is associated with the faultVariable attribute of the <throw> activity. The faultVariable attribute specifies the variable name which is represented as a data type in the Process DAML. The <throw> activity will be mapped onto a DAML-S atomic process with one output of same data type as of the fault variable specified in the <throw> activity. There will be no inputs for the atomic process derived as this activity is used to generate faults and send fault messages. The derived atomic process will be of its own type and will not inherit from any atomic processes derived from WSDL as the <throw> activity does not specify the port type and operation for communication. In the case of the faultVariable attribute not specified, the data type of the output will be a DAML object (i.e. Thing) by default.
3.5. Structured Activities

BPEL4WS structured activities describe how a business process is created by composing the primitive and structured activities. They prescribe the order in which a collection of activities (both primitive and structured) takes place and expresses the control patterns, data flow, handling of faults and external events. The structured activities include: ordinary sequential control between activities which are provided by <sequence>, <switch> and <while>; nondeterministic choice based on external events which are provided by <pick>; Concurrency and synchronisation between activities are provided by <flow> [1]. DAML-S composite processes are composed of subprocesses (atomic or composite) and share the same concept of BPEL4WS structured activities. Thus, the structured activities in BPEL4WS will be recursively mapped onto DAML-S composite processes. Every composite process has a control construct associated with it. The control constructs (sequence, choice, repeat-while, etc) are closely related to BPEL4WS structured activities. The inputs and outputs of composite processes are derived from the corresponding inputs and outputs of atomic subprocesses and will be computed normally (explained in sub-section 3.6).

The <sequence> activity contains one or more primitive or structured activities that are performed sequentially based on the order as listed within the <sequence> activity. The <sequence> activity completes when the final activity listed in the <sequence> activity gets completed [1]. A DAML-S Composite process with control construct of type Sequence will be used for this mapping. It will list the sub-processes (atomic or composite) which will be performed in the order in which they are listed. The components of the derived composite process can be either atomic processes or composite processes depending upon the internal activities in <sequence>. All the sub-processes of the derived composite process are recursively declared.

The <switch> activity supports the conditional behaviour in the pattern that occurs frequently. The activity consists of an ordered list of one or more conditional branches defined by case elements, followed optionally by an otherwise branch. The activity of the branch whose condition holds is performed and the <switch> gets completed. If no branch condition holds then the activity of the otherwise branch is performed [1]. A DAML-S Composite process with control construct of type Choice will be used for this mapping. The <while> activity supports the repeated performance of a specified iterative activity in the business process. The iterative activity is performed until the given condition no longer holds [1]. A DAML-S composite process with control construct of Type Repeat-While will be used for this mapping. Repeat-While allows a sub-process to repeatedly iterate until the whileCondition is true.

The most fundamental semantic effect of a grouping set of activities in <flow> is to enable concurrency. A <flow> activity completes when all the activities in the <flow> have completed. More generally, a <flow> activity creates a set of concurrent activities directly nested within it. It further enables expression of synchronisation dependencies between activities that are nested directly or indirectly within it. The link construct and standard elements (source and target) are used to express these synchronisation dependencies [1]. A DAML-S composite process with Concurrent-Sync control construct will be used for this mapping. A DAML-S Concurrent-Sync control construct is a sub-class of the Split-Join control construct which allows concurrent execution of a collection of sub-processes, with barrier synchronisation. Complete execution of all the sub-processes in the Concurrent-Sync is required to complete the process the same as the <flow> activity in BPEL4WS.

3.6. Data Flow

A Data Flow DAML file will be produced as one of the outputs for this whole mapping process. The Data Flow DAML contains the process annotations for the Process DAML which will relate various process parameters to each other as defined in Process DAML. It will separate the data flow from the process definition. The inputs and outputs of the atomic processes derived from <receive>, <reply> and <invoke> activities accordingly will be described in the Data Flow DAML using the DAML-S valueOf class to refer to their respective super class atomic process’ inputs and outputs.

When composing atomic processes into composite processes (i.e. when a composite process has atomic sub-processes in it), it is crucial that the inputs and outputs of the sub-processes are related to each other. This is addressed using the DAML-S valueOf class (representing that two parameters used for referencing are equal), used similarly as above for referencing the inputs and outputs of the derived atomic processes from primitive activities to their corresponding super class atomic process’ inputs and outputs. Furthermore, we represent the referencing of the inputs and outputs of the derived composite processes from structured activities in the Data Flow DAML using the DAML-S valueOf class thus representing the complete data flow for both atomic processes and composite processes derived from BPEL4WS primitive and structured activities respectively.

4. BPEL4WS2DAML-S v1.0

In this section, we will discuss our GUI mapping tool (BPEL4WS2DAML-S v1.0) which maps the BPEL4WS specification onto the DAML-S based workflow process.
ontology. It implements the complete mapping specification and uses the same inputs (BPEL and WSDL) and produces the same outputs (Process DAML and Data Flow). Examples as well as the tool can be downloaded from our website http://www.it.swin.edu.au/centres/cicec/bpel2damlS.htm. We provide the reusable packages and the classes (Java SDK 1.4.1 or higher) as well as open sources upon request.

Figure 2. WSDL meta model

Figure 3. BPEL4WS meta model

Figure 2 illustrates the Meta model of WSDL and Figure 3 illustrates the Meta model of BPEL4WS. The top level Meta model of DAML-S ontology can be found in [8]. Some of the main features of the tool are described next. Object Explorers are used to represent the object view of the input (BPEL and WSDL) and the output (Process DAML) files. They allow easy navigation and representation of hierarchical inheritance and relationships between the objects and their properties using a tree structure. A Project Validator is used to associate correct Service Description (WSDL) for a given BPEL4WS process in mapping. The tool provides help with brief description for the important functionalities of the tool and how to interact with the tool. Users can also simultaneously view the tree structure representing the object view of the source files (inputs and outputs) and also the BPEL, WSDL and DAML files in Microsoft Internet Explorer.

However, the tool does not support multiple service descriptions for a given BPEL4WS process. All the WSDL definitions must be defined in a single service description File (WSDL input file). The tool also does not provide very comprehensive feedback though sufficient information provided in error messages, and syntax checking of the source files.

5. Related Work

The derived mapping specification in this paper is based on and strictly conforms to the following language specifications and standards used in this effort: WSDL v1.1 – March 2001, BPEL4WS v1.1 – May 2003 and (DAML-S) v0.9 – May 2003. Our tool represents the implementation of this mapping specification. Our work greatly extends the work reported in [14]. To date it has attracted many responses and we believe it will contribute to the evolution of process ontology models, i.e. OWL-S, which has just reached version 1.0 in November 2003 [6].

The derived mapping specification lacks crucial concepts that facilitate the composition of complex Web services and business processes while implementing using the DAML-S process ontology [9, 12]. These shortages need to be challenged in order to represent complete workflow and business logic of BPEL4WS in the DAML-S process ontology, as what has been aimed at in [10]. In doing so, we also need to consider the limitations of our tool in the near future, in order to improve the efficiency and flexibility in mapping.

Table 2 below shows the BPEL4WS elements and activities that cannot be mapped onto the DAML-S workflow process ontology together with the reasons (based on DAML-S v0.9 specification).

<table>
<thead>
<tr>
<th>Elements / Activities</th>
<th>Reasons</th>
</tr>
</thead>
<tbody>
<tr>
<td>faultHandlers</td>
<td>DAML-S does not support fault/error handling and any recovery protocols</td>
</tr>
<tr>
<td>compensationHandler</td>
<td>DAML-S does not define any recovery protocols</td>
</tr>
<tr>
<td>eventHandlers</td>
<td>DAML-S does not support raising of and listening to events</td>
</tr>
<tr>
<td>partnerLinks</td>
<td>DAML-S does not specify the relationships between the participants of the process.</td>
</tr>
<tr>
<td>&lt;wait&gt;</td>
<td>DAML-S does not specify how to block and wait for synchronisation</td>
</tr>
<tr>
<td>&lt;terminate&gt;</td>
<td>DAML-S does not specify how to explicitly terminate a DAML-S process</td>
</tr>
<tr>
<td>&lt;assign&gt;</td>
<td>DAML-S does not support the use of variables and assignment statements.</td>
</tr>
<tr>
<td>&lt;compensate&gt;</td>
<td>DAML-S does not specify how to compensate in the case of any errors in an activity</td>
</tr>
<tr>
<td>&lt;scope&gt;</td>
<td>No suitable control construct specified in DAML-S for this structured activity</td>
</tr>
</tbody>
</table>

Furthermore, while mapping the <flow> activity to the DAML-S Concurrent-Sync composite process (discussed above); we cannot represent the synchronisation of and links between the sub-processes in the derived composite
process due to the lack of waiting and synchronisation features in DAML-S v0.9. Thus, many issues of crucial BPEL4WS language constructs need to be considered in future mapping to achieve the accuracy and reliability in mapping of the BPEL4WS specification to the DAML-S based workflow process ontology.

We are also implementing mapping from XPDL to new versions of DAML-S, that is, OWL-S. Hopefully, with the upgrade to BPEL2OWL-S in the near future, the perplexing semantics problems, which are haunting around the mapping from the Web service specification to process ontology, will be better resolved.

6. Conclusions

With the ability of DAML-S to define the Web services content vocabulary in terms of objects and complex relationships between them including classes, sub-classes, cardinality restrictions, hierarchical inheritance, and with the ability to provide a shared set of terms describing the application domain with a common understanding for sharing information and knowledge and with well defined semantics, using DAML-S as an ontology in our project overcomes not only the limitations and weaknesses of BPEL4WS but also some of the XML limitations, and solves data integration and interoperability problems and issues faced today in the Web services world.

In this paper, we have described the partial mapping of the BPEL4WS specification to the DAML-S based workflow process ontology by extending the work done in WSDL to DAML-S and developing a GUI mapping tool to support this mapping. By overcoming the limitations and weaknesses in our derived mapping specification from BPEL4WS to DAML-S in the near future, we believe that we can achieve the accuracy, efficiency and reliability in our mapping and will be able to represent a complete and accurate business logic of a BPEL4WS process in the DAML-S based workflow process ontology. Both the language specifications (BPEL4WS and DAML-S) are evolving. This will allow us to easily track changes in the specifications of these languages to improve our mapping between these specifications.

Acknowledgements

This work is supported by Swinburne Vice Chancellor’s Strategic Research Initiative Grant 2002-2004, as well as the National Natural Science Foundation of China under grant No.60273026 and No.60273043. We gratefully acknowledge invaluable feedbacks from related research communities.

References: