Semantic Web Service Engineering for Semantic Business Process Management

Mick Kerrigan
STI Innsbruck,
University of Innsbruck,
Austria

Barry Norton
Knowledge Media Institute,
Open University,
United Kingdom

Elena Simperl
STI Innsbruck,
University of Innsbruck,
Austria

Dieter Fensel
STI Innsbruck,
University of Innsbruck,
Austria

Abstract
The Semantic Business Process Management (SBPM) approach from the SUPER project utilizes a Semantic Execution Environment (SEE) for the automatic discovery, composition, mediation, and invocation of Web services. In order to enable the Semantic Execution Environment, an engineer must create semantic descriptions of functional, non-functional, and behavioural aspects of Web services and end-user requirements. In this paper we take the first step into the emerging field of Semantic Web Service engineering by identifying a number of application scenarios within which Semantic Web Services can be used and identifying the different engineering activities that the engineer must perform in order to enable these scenarios. Information was elicited directly from those who are actively developing Semantic Web Services by the use of survey. Thus the scenarios are built based on direct input from a cross section of the Semantic Web Service community. The results in this paper act as a starting point for the Semantic Web Service engineering methodology that we are currently in the process of developing.

Introduction
Business Process Management (BPM) is an established discipline whereby the processes of a company are modelled, monitored, managed, and adapted according to business experts’ viewpoint, well-separated from the lower-level IT concerns associated with their realisation. Meanwhile the approach of Service-Oriented Architecture (SOA) has made strides towards supporting the requirements of agile cross-organisational business processes at this implementation level. Semantic Business Process Management (SBPM) (Hepp et al. 2005) is a recent approach based on the application of ontology-based semantics to bridge the gap between the business analyst’s and IT department’s viewpoints in BPM, which is an application of the principle of ontological role separation. Many SBPM approaches use Semantic Web Services (SWS) and semantics-driven execution.

Several aspects of the Web Services Modeling Ontology (WSMO) (Fensel et al. 2006) are of particular advantage in this regard. Firstly the service-level application of ontological role separation, distinguishing goals from the descriptions of services that meet them, is useful in the separate description and mediation between abstract business process models and executable business processes. Secondly the use of explicit mediators in linking descriptions scales up well to the process level. Finally the execution support provided by WSMO-compliant Semantic Execution Environments (SEEs) (Fensel, Kerrigan, and Zaremba 2008) integrates well with existing support for business process execution (Nitzsche et al. 2007), as seen in the SUPER project.

In order to achieve this meeting between different communities beyond the sphere of research it is necessary to concretely and methodically describe the results and shared understanding of 7 years research and development in SWS. In particular the conceptual models, languages, frameworks, and tools must be placed in the context of methodologies, best practices, and guidelines. However in order to go about creating such methodological support, which does not currently exist, it is important to understand the high level engineering activities that must be conducted by the SWS Engineer. Thus in this paper we present a set of SWS Engineering scenarios that describe the different activities that must be conducted by the SWS Engineer in order to enable each of these scenarios. These scenarios have been built based upon an analysis of recent SWS and SBPM projects along with a survey conducted with experts in the SWS community.

The scenarios provide a starting point for the authors’ ongoing research into methodologies for SWS Engineering by providing a coarse grained overview of the engineering activities that must be supported by such a methodology. They can also be used by those wishing to implement an application that utilizes one or more broker services from a SEE to understand which engineering activities need to be performed in order to realize a particular scenario. As the scenarios can easily be combined this ensures that even when implementing complex applications the engineer can be sure that they understand exactly which artifacts need to be engineered to enable the SEE broker services to function correctly. The scenarios can also be utilized by those in the SWS community who are trying to provide SEE implementations, or by SBPM practitioners aiming to apply such implementations, as they provide an overview of the artifacts that SWS Engineers will use to interact with their systems, and thus provide a baseline of functionality for a SEE.
Background

In order to support the separation of business analysts' and IT viewpoints of business processes the SUPER project has introduced two main ontologies: the Business Process Modelling Ontology (BPMO)\(^2\) and a Semantic BPEL model, which is grounded for execution to a compliant extended WS-BPEL 2.0 schema, BPEL4SWS (Nitzsche et al. 2007). Common features between the two can be mediated using ontology-based rules (Norton, Cabral, and Nitzsche 2009).

BPMO’s stated intention is to “[model] business processes at the semantic level, integrating knowledge about the organisational context, workflow activities and Semantic Web Services.” The workflow aspect is conceptualised via the abstractions encoded in Workflow Patterns (van der Aalst et al. 2003) and aims to be graphically represented in a fragment of the Business Process Modelling Notation (BPMN) (Object Management Group 2006). The patterns covered are also intended to abstract over the features of Event-driven Process Chains (EPCs), the extended model of which in the ARIS toolset is inspiration for the modelling of organisational context (Scheer, Oliver, and Otmar 2005).

The Web Service Modeling Ontology (WSMO) (Fensel et al. 2006) is a conceptual model for Semantic Web Services and has four top level elements, namely Ontologies, Web Services, Goals and Mediators. Ontologies are the basis for the other descriptions by providing the terminology that they use. WSMO Web Services provide a semantic description of both the function of a service, in terms of a Capability, and the mechanism for interacting with it, in terms of an Interface. A WSMO goal allows for the requirements of the requester to be semantically described. Finally, WSMO Mediators provide a means to resolve heterogeneity issues that inevitably occur between the other elements due to the open and distributed nature of the Web.

WSMO’s service model is primarily used in two regards in BPMO. The concept of goal allows the requirements for external tasks to be functionally specified, along with non-functional requirements related to, for instance, Quality of Service. The concept of mediator allows both the specification of necessary data mediation within a process, as well as a mediation process to be specified between a set of processes that are otherwise lacking in mutual conformance. In translation to Semantic BPEL mediators are intended to play the same role (Nitzsche and Norton 2008). Goals, on the other hand, may either be left in place for run-time matching to a web service, as described below, or may be replaced with the semantic description of a suitable service in the executable process.

Requirements Gathering

In order to try to understand how engineers are currently going about the process of engineering Semantic Web Services, we distributed a survey to elicit information from those that are actively engineering Semantic Web Services.

Along with this direct contact approach we also gathered information indirectly through deliverables generated in the course of research projects mentioned by individual survey respondents. The survey, entitled “Experiences with Semantic Web Services - A Practitioners Perspective”\(^3\), was designed around the different phases of the Software Life Cycle in order to gather information on each of these phases and to keep participants focused on the particular aspect of the process of Semantic Web Service Engineering while they are conducting the survey. Questions were predominantly open answer questions along with some multiple choice questions, using the Semantic Web Service terminology from the WSMO community. The survey was completed by 51 participants covering 21 nationalities spanning 16 different organizations in 10 different countries, including STI Innsbruck, DERI Galway, the Open University, EPFL, Ontotext, University of Stuttgart, and SAP Research. 59% of participants said they were experienced with Semantic Web Services, 25% said that they have medium experience, while 16% classified themselves as novices. One of the limitations of the survey is that due to the infancy of the Semantic Web Service area, 86% of the participants come from academia, with the remaining 14% coming from industry. It should also be noted that 33% of all participants gained experience with Semantic Web Services while working in projects related to Semantic Business Process Management. Based on the results of this survey, and analysis of relevant project deliverables, the scenarios in the next section are identified that show the different ways in which Semantic Web Services are currently being engineered.

Semantic Web Service Engineering Scenarios

Systems developed by following the Semantic Business Process Management approach do not necessarily conform to any one single architecture at the (Semantic Web) Services level. One system may use one set of the broker services from a Semantic Execution Environment to automate parts of the process of using Web services, while another system may use completely different broker services and thus have very different behaviour. The types of Semantic Web Service artifacts that must be created in any particular case are thus highly dependent on the types of broker services that are used in that system. Based on the survey results and a study of relevant Semantic Web Service projects, this section explores the different scenarios in which Semantic Web Services are currently used. These scenarios can be used by engineers to design systems that use Semantic Web Services and to understand the different artifacts that must be created in order to enable such systems. During the design phase for a new system, the service engineer can identify the relevant scenarios that are required to realize the requirements for that system. The engineer can combine these scenarios in order to design a rich and complex system that utilizes Semantic Web Services. Building a design from these scenarios allows the engineer to understand the artifacts that must be implemented in order to enable the broker services


\(^3\)http://survey.sti2.at/public/survey.php?name=sws_mick
used in the final system. Each of the scenarios are described in detail below.

**Scenario 1: Using Semantic Web Services for Service Advertisement**

One of the key problems with SOA, even within one single organization, is that of service management, i.e. how to know what services already exist, what their functionalities are, and where to access them when transitioning a business process model into a service-based implementation. Traditionally UDDI registries, were used to keep track of services that already exist, with both public and private registries being used to enable reuse of existing services. When a particular functionality is desired within a SOA based system, an engineer would search the available registries for services fulfilling that functionality, incorporating those found into the system or developing new services if no services with the required functionality exist. Semantic Web Services can be used to enable the creation of an intelligent service repository, similar to the notion of a UDDI registry except that the engineer need not manually search through textual descriptions in order to find services that can perform some particular desired functionality. Instead a discovery engine can be used to match the functionality advertised in the available Semantic Web Services with the end users Goal, which describes the functionality desired by the end user. Discovery engines can also be coupled with a ranking engine such that a user centric ranking of services can be provided by the intelligent service repository, with such ranking based on the described or observed non-functional behaviour of the service. There are two common approaches to service discovery, with these approaches having different requirements on the artifacts that need to be developed. These approaches are described in the scenarios below:

**Scenario 1a: Capability-based Service Advertisement**

Discovery engines that use capability-based service advertisement find provider services that can fulfill an end user Goal by matching descriptions of the required functionality desired by the end user and the offered functionality of provider Web services. In this scenario the service provider provides a capability description of the service being advertised. In order to query the repository the service requester provides a capability description of the required service within a goal. The actual design and structure of these capabilities depends upon the discovery approach used by the discovery engine, e.g. keyword (Keller et al. 2005a), lightweight (Keller et al. 2005b), or heavyweight (Keller, Lausen, and Stollberg 2006). These capability descriptions can use ontologies that have already been created to ensure that the capabilities are formal and unambiguous in nature. If the intelligent service repository wants to return a ranked list of services that fulfill the Goal, there is also the need for the provider to annotate the service with non-functional properties that describe Quality of Service (QoS) aspects of the service. The requester should then provide information on the non-functional properties that are relevant to him within the goal description such that the repository can return a personalized ordering.

**Scenario 1b: Mediator-based Service Advertisement**

In contrast, discovery engines that use mediator-based service advertisement use WSMO mediators to define matches between Web Services and Goals. This approach is often referred to as static brokerage (Cabral et al. 2006) as the matches between Goals and Web services are defined in advance and thus the process of discovery is a simple lookup of a mediator. These mediators in some approaches are supported by a collection of rules that defines the mediators behaviour, for example in (Carenini et al. 2008). In some cases a capability-based discovery engine is used at design time in order to find the matches, which are then encoded as mediators. Mediator-based discovery is especially suitable in an environment where the number of services are limited and there is a desire to only use particular services. For example, in cases where service level agreements and written contracts are required prior to using a particular providers’ services. To enable this scenario the service provider defines a service and a goal description. These descriptions are then linked together by the definition of a Web Service to Goal mediator (wgMediator). In some cases the provider will not create a new Goal, but instead reuse an existing Goal that was created by a provider or requester.

In order to query, a service requester instantiates a particular goal description made available by a given provider and a match occurs if there is a wgMediator between the instantiated Goal and some services. Mediator-based service discovery can be combined with Scenario 1a, such that if multiple services fulfil a specific Goal they can be differentiated by their functional descriptions. In such a combination the mediator-based discovery acts as a prefilter to capability-based discovery, e.g. restricting the services that can be discovered to a set of approved services or just reducing the search space to a more manageable size. Also Non Functional properties as defined in Scenario 1a may be used within this scenario to differentiate services with the same functionality from one another. In some cases the wgMediator between the Web Service and Goal will be annotated with the relationship between them, i.e the Web Service can be an exact, plug-in, or subsumption match with the Goal.

An extension to this approach, used by (Stollberg, Hepp, and Hoffmann 2007), enables more relationships between Web Services and Goals to be specified. In this extension the service provider attaches the created service to a pre-existing hierarchy of services, and attaches the created Goal to a pre-existing hierarchy of Goals. These hierarchies are built by defining Web Service to Web Service mediators (wwMediator) between different Web Service descriptions, and Goal to Goal mediators (ggMediator) between different Goal descriptions. Attached to these mediators are a definition of the relationship between the Goals or Web Service descriptions, for example a particular Goal is a specialization of another Goal or a particular Web Service is a generalization of another Web Service. Thus a wgMediator defined between a Web Service description and a Goal description signifies not only a relationship between that Web Service and that Goal, but also between the more general or specific Goals and Web Services in each of the hierarchies.
Scenario 2: Using Semantic Web Services for Service Invocation

In a different context the services needed within a system may already be known and implicit in the business process model, so there is no need for any form of discovery; however there may be a need to automatically invoke these known services. This can be particularly important where the interfaces of the services change regularly. In a traditional scenario the requester would need to change the SOA based system whenever these interfaces are changed; however with Semantic Web Services it is possible to semantically describe the interface of the service such that it can be automatically invoked. Thus whenever the interface changes the provider changes the semantic interface description and the requesters system continues to automatically invoke it, except in a different manner according to the new semantic description. Scenario 1 and 2 can be combined within a system to enable services to be discovered based on their advertised functionality and then automatically invoked based on their semantic interface description. There are two ways in which service invocation can occur and these are described in the following scenarios:

Scenario 2a: Service Choreography-based Invocation

To enable requesters to automatically invoke their service the service provider needs to provide an unambiguous and formal description of the behaviour of that service. The process model, entitled a choreography, describes how the requester should interact with the service in terms of the sequence in which messages should be sent to the service. Thus to enable this scenario the service provider should create a choreography description of the service. Along with the choreography the provider should create a grounding that enables ontological instances to be sent to the service according to a particular mechanism. For example with a SOAP Web Service, Ontological instances must be lowered to XML data conforming to the services XML schema and XML data received from the service must be lifted to ontological instances conforming to the services ontologies. To perform the invocation of the service the requester need only provide the ontological instances as described within the services choreography.

Scenario 2b: Goal Choreography-based Invocation

The simple scenario described in Scenario 2a is only possible in cases where all the information needed to execute the service choreography is available prior to the execution. In some cases the requester will need to generate new data based upon the responses from the providers service. Thus in such a scenario the requester also requires a behavioural description such that a conversation between the requester and service can be made. To enable this scenario the requester must provide a choreography description within the Goal that describes how they want to interact with the service. Along with this choreography a grounding may also be provided such that the requester can interact with the service in terms of a specific syntactic representation. At run-time mismatches between the choreographies of the Goal and Web Service descriptions can often be automatically resolved through the use of process mediation. In this scenario the provider acts in the same manner as in Scenarios 2a and 2b, intra- or extra-organisationally, and manually creates a composition of services such that it is available for execution at runtime. To do this a new Semantic Web Service description needs to be created that is not associated with a specific Web service as in Scenario 2, but is realized by a number of Semantic Web Services and Goals. To enable this scenario an engineer needs to create an orchestration that brings together the individual Semantic Web Services and Goals in the composition to create the desired functionality of the new composite service. The orchestration may also contain mediators where heterogeneity issues need to be resolved between different Web Service and Goal descriptions in the orchestration. From this orchestration the engineer should project out a choreography for this new service and add it to the description. In this scenario the service requester acts in the same manner as in Scenario 2, treating this new composed service as if it were a standard Semantic Web Service. If there is a requirement for this new composite service to be discovered then this scenario can be combined with Scenario 1 in order to create an advertisement for it.

Scenario 3: Using Semantic Web Services for Service Composition

Scenario 1 enables the development of systems that can automatically discover existing services so that they can be incorporated into SOA based systems; however it is often the case that tasks at the business process level that can be captured as single coherent Goals cannot be fulfilled by a single service at the IT level, i.e. no single service exists with the same functionality as that described at the business process level. In such a case it may be possible to combine a number of discrete services in order to fulfil the requesters requirements. There are two approaches to service composition, namely design-time and run-time based composition with each of these approaches having different requirements on the Semantic Web Service artifacts that need to be developed by requester and provider. These two approaches are described in the scenarios below:

Scenario 3a: Design-time Service Composition

The simplest form of composition occurs when some party decides to act as a virtual provider, intra- or extra-organisationally, and manually creates a composition of services such that it is available for execution at runtime. To do this a new Semantic Web Service description needs to be created that is not associated with a specific Web service as in Scenario 2, but is realized by a number of Semantic Web Services and Goals. To enable this scenario an engineer needs to create an orchestration that brings together the individual Semantic Web Services and Goals in the composition to create the desired functionality of the new composite service. The orchestration may also contain mediators where heterogeneity issues need to be resolved between different Web Service and Goal descriptions in the orchestration. From this orchestration the engineer should project out a choreography for this new service and add it to the description. In this scenario the service requester acts in the same manner as in Scenario 2, treating this new composed service as if it were a standard Semantic Web Service. If there is a requirement for this new composite service to be discovered then this scenario can be combined with Scenario 1 in order to create an advertisement for it.
to guide the process of discovering services that can be included in the composition. The requester can optionally provide a choreography within the Goal that dictates the external visible behaviour of the new composite service. The requester can also optionally provide a partial orchestration that can restrict the orchestration of the composed service that will be created, e.g. specifying that some specific service must be used for a specific task in the orchestration or defining the overall flow of the orchestration with Goals as placeholders. Finally if the requesters wants to interact using XML, a grounding in terms of lifting and lowering mappings should be provided to transform ontologies instances to and from XML.

Scenario 4b: Reusing Existing Ontologies

One of the easiest ways to create ontologies for describing Web services is by reusing existing ontologies. The top down approach existing standard ontologies are directly reused to describe the Web service. This results in very few heterogeneity issues existing, but the process of creating a grounding becomes more complicated due to the potential gap between the services XML Schema and the reused ontologies. This approach is especially useful in relation to Scenario 1 where no grounding is required, and to SBPM in general.

Scenario 4c: Reengineering Existing Ontologies

The middle-out approach is a compromise approach between the approaches described in Scenarios 4a and 4b. In this approach existing ontologies are reengineering taking into account the Web services XML Schema. In this approach creating a grounding is more complex than in Scenario 4a and more heterogeneity exists than in 4b; however both are manageable. This approach is very useful in a system using a mixture of Scenarios 1 and 2 and can be of some use of the upper level of SBPM, but care must be taken not to allow the mixing of concerns to compromise the abstractions of the business analysis level.

Scenario 5: Enabling Interoperability between Ontologies

Using the ontology engineering approaches described in Scenarios 4a and 4c results in ontologies that are locally relevant but not shared with others in the community. If a given Web service is described using these ontologies it will not be possible for a requester to get a match between the Goals in their business processes and these services unless they use the same ontologies. Therefore to enable matches between requester Goals and providers Web Services in a heterogeneous ontological environment it is necessary to define ontology to ontology mediators (ooMediator) between local provider ontologies and standard ontologies on the Web. This heterogeneity can also exist in Scenario 4b, in cases where there is a fragmentation within a community and a number of standard ontologies for describing the same domain exist. Therefore it is also necessary to define ontology to ontology mediators between these standard ontologies to ensure interoperability between requester Goals and provider Web Services using these different ontologies. From the survey we discovered that some form of ad hoc standardization on using ontology to ontology mappings for enabling ooMediators has occurred. Thus an ooMediator is usually accompanied by a mapping document containing mappings between the different elements in the source and target ontologies of the ooMediator, as described in (Mocan and Cimpian 2007).

Conclusions and Future Work

The scenarios described in this paper were built based on feedback from those who are actively engineering Semantic Web Services, including in the context of SBPM, and therefore provide an insight into how Semantic Web Services are actually being used by the community. The matrix in figure 1 provides an overview of the different artifacts that the engineer needs to create in order to enable each of the scenarios, as described in the previous section. When combining the different scenarios the engineer need only combine the necessary columns from this matrix in order to understand which artifacts are needed to realise the application.
In terms of future work, we intend to use these scenarios to build a methodology for Semantic Web Service Engineering, specifically taking the SBPM context into account. In this methodology we will define the different activities that need to be performed to successfully gather requirements, create a design, implement, test, deploy, and document the different artifacts identified in the different scenarios. We also intend to enhance the functionality of the Web Service Modeling Toolkit (WSMT) (Kerrigan et al. 2007), an integrated development environment for Semantic Web Services under development for the last four years, to support the methodology and the activities within it.

Table: Overview of different artifacts by scenario

<table>
<thead>
<tr>
<th>Scenario</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
<th>3a</th>
<th>3b</th>
<th>4a</th>
<th>4b</th>
<th>4c</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capabilities</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF Props &amp; Prefs</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choreographies</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orchestrations</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ontologies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groundings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediators</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OO Mappings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Overview of different artifacts by scenario

Acknowledgements

The work is funded by the European Commission under the projects ACTIVE, COIN, LarKC, MUSING, Salero, SEEMP, SemanticGov, Service Web 3.0, SHAPE, SOA4ALL, SUPER, SWING, TripCom and by the Austrian FFG (Österreichische ForschungsförderungGesellschaft mbH) under the projects myOntology, SARID, and SARIT.

References


