
Foundations for specifying OWL-S groundings

Gerald C. Gannod*

Department of Computer Science and Systems Analysis,
Miami University,
Oxford, OH 45056, USA
Fax: +1-513-529-1524
E-mail: gannodg@muohio.edu

*Corresponding author

Raynette J. Brodie and John T.E. Timm

Department of Computer Science and Engineering,
Arizona State University,
Tempe, AZ 85287-8809, USA
E-mail: raynettejb@gmail.com E-mail: area51@asu.edu

Abstract: OWL-S is an ontology in the Web Ontology Language (OWL) that is used to describe and specify semantic web services. While OWL-S provides a promising mechanism for specification, publication, discovery, integration and access, the learning curve can be steep. As such, tool support for generating OWL-S specifications is a necessary requirement for effectively using the language for complex service orchestration. One of the advantages of OWL-S is its flexibility. Specifically, it facilitates the specification of many groundings (e.g. bindings) to a semantic web service. In this paper, we characterise groundings, describe an approach for generating groundings for a semantic web service and demonstrate how the use of lightweight interactive tools can facilitate creation of groundings for a semantic web service.

Keywords: semantic web services; OWL-S; groundings.

Reference to this paper should be made as follows: Gannod, G.C., Brodie, R.J. and Timm, J.T.E. (2007) 'Foundations for specifying OWL-S groundings', *Int. J. Business Process Integration and Management*, Vol. 2, No. 1, pp.49–59.

Biographical notes: Gerald C. Gannod is an Associate Professor in the Department of Computer Science and Systems Analysis at Miami University and an Adjunct Professor in the Department of Computer Science and Engineering at Arizona State University. He received the MS (1994) and PhD (1998) in Computer Science from Michigan State University. His research interests include service-oriented computing, software reverse engineering, software product lines, formal methods for software development and software architecture.

John T.E. Timm is working for his PhD in the Department of Computer Science and Engineering at Arizona State University. He received a BSE in Computer Systems Engineering and a MCS in Computer Science from Arizona State University. His research interests include: service-oriented computing, model-driven architecture, semantic web services, software engineering and embedded systems. He has eight years of professional experience as a Software Engineer. Currently, he is working with Gerald Gannod at Miami University and Susan Urban at Arizona State University in the area of semantic web service specification and execution.

Raynette J. Brodie is a Software Developer with The 41st Parameter. She received a BS in Computer Science from Arizona State University. Her research interests in the area of software engineering and service-oriented computing. The work described in this paper was performed while Brodie was a student at Arizona State University.

1 Introduction

A semantic web service extends the capabilities of a web service by associating semantic concepts to the web service in order to enable better search, discovery, selection, composition and integration. The OWL-S specification language (Martin et al., 2005) has been created to provide a mechanism for describing domain concepts using ontologies

while providing constructs for specifying information about how to compose and access services. While the benefits of semantic web services are attractive, they have yet to gain widespread adoption due to the lack of tools that support their specification and development. The current state of the practice for non-semantic web services is to develop the service using tools such as Visual Studio .NET and Apache Axis and then to automatically generate an appropriate

WSDL specification. Imposing further requirements of adding OWL-S specifications in order to create semantic web services offers many challenges including a learning curve that is often undesired by developers.

One of the advantages of OWL-S is its flexibility. Specifically, it facilitates the specification of many groundings (e.g. bindings) to a semantic web service. We are developing an approach for constructing OWL-S specifications that focuses on two activities. Firstly, we have developed an approach based on the use of Model Driven Architecture (MDA) (Miller and Mukerji, 2003) for generating OWL-S profile and process specifications (Gannod et al., 2006). Secondly, we have developed an interactive approach for generating OWL-S groundings. This two part approach allows development of semantic web services to occur in two stages. The semantic and architectural concerns associated with specifying semantic web services can be performed by software architects and knowledge architects. The mapping of web services described using WSDL to operations contained in the profile and process specifications are intended to be performed by developers of web services as the services are constructed or by architects as they identify existing services that meet the intended behaviours of a semantic service. In this paper, we present a characterisation of OWL-S groundings and describe our approach for facilitating the creation of groundings through the use of an interactive tool.

The rest of this paper is organised as follows. Section 2 describes background information relevant to semantic web services. Section 3 describes our general framework as well as the details of an interactive tool for constructing semantic web services. An illustrative example using our approach is presented in Section 4. Related work is discussed in Section 5 and finally, Section 6 draws conclusions and suggests future investigations.

2 Background

This section describes background information on web services and OWL-S.

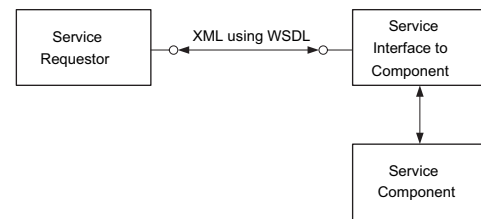
2.1 Web services

A web service is a modular, well-defined software component that exposes its interface over a network. Applications use web services by sending and receiving XML messages over HTTP as shown in Figure 1. A web service provides access to a service component via an interface

defined by a Web Service Description Language (WSDL) (Christensen et al., 2001) specification. Web services provide the foundation for loosely coupled, service-oriented software systems. They allow multiple organisations to interact in a uniform, well-defined manner, and are a major step towards interoperability between heterogeneous distributed systems.

Web service interfaces are defined using WSDL. WSDL is an XML-based language for describing the interface of a web service including message types and bindings. While WSDL does not support the use of formal specifications of the semantics of a web service, current proposals for WSDL-S (Akkiraju et al., 2005) are moving in that direction.

Figure 1 Web service interactions

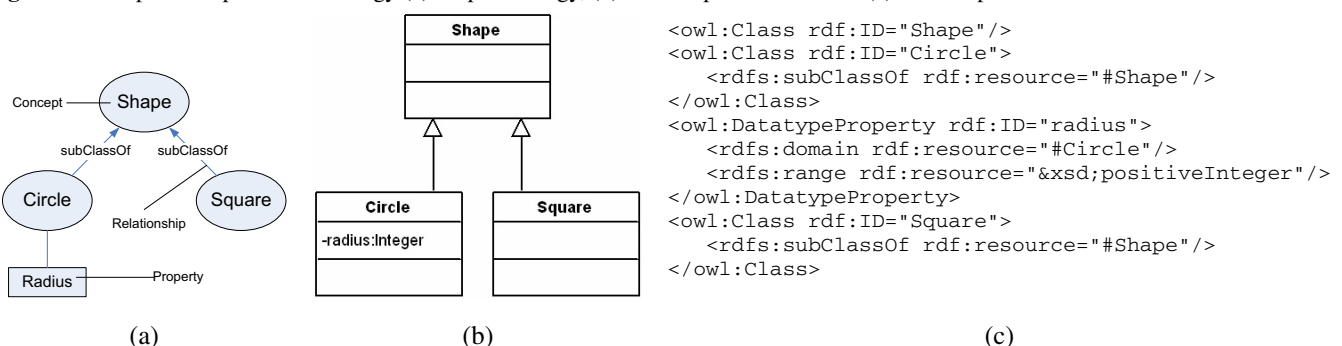


2.2 Ontologies and semantic web services

An ontology is a set of concepts, their properties, and the relationships between them. Ontologies provide the building blocks for expressing semantics in a well-defined manner (Berners-Lee et al., 2001). An example of a shape ontology is shown in Figure 2(a). The example shows many of the different aspects of ontologies, including *concepts* and *relationships*. A concept describes some significant semantic element or instantiable object. In the example, *Shape* constitutes such an element or concept. Concepts can be subclassed in order to show specialisation and generalisation. Properties describe attributes of concepts in order to help partition a domain.

Ontologies serve as the primary building block for adding semantics to web services. Modelling with ontologies is similar to domain modelling in the software engineering context (Kiko and Atkinson, 2005). Several analogies exist between software modelling and knowledge engineering including the use of classes, inheritance and properties. For instance, a corresponding UML rendering of the ontology shown in Figure 2(a) is shown in Figure 2(b). The corresponding OWL XML representation is shown in Figure 2(c).

Figure 2 Simple example of an ontology (a) shape ontology, (b) UML representation and (c) OWL representation



Efforts to define the *ontology definition metamodel* (OMG, 2005) have been undertaken to bridge the gap between the knowledge engineering and software engineering communities by defining mappings from ontologies to the relevant visual metaphors in the UML. These goals meet our own in supporting the specification of OWL-S services using UML and MDA.

The Web Ontology Language (OWL) is an XML-based language for describing ontologies (Smith et al., 2004). OWL was designed to allow for the specification of semantic descriptions for resources on the web in order to support interpretation by autonomous software agents. These resources can be anything from simple web pages to web services. Because the syntax of OWL is XML, it is platform-independent, can be easily transferred over a network and manipulated with existing automated tools.

Semantic web services are web services that have been described semantically, enabling automated discovery, composition and invocation. Semantic-rich languages are used to describe these services, their capabilities and expected inputs. These languages, such as the Resource Description Framework (RDF) and OWL, are used to create ontologies of concepts used in service descriptions. The process of semantic web service specification involves using ontologies to give meaning to a service and its components.

OWL-S (Martin et al., 2005) is an upper ontology for web services written in OWL. The OWL-S ontology is intended to provide a base for creating semantic descriptions of services. A competing semantic web service infrastructure is the Web Service Modelling Ontology (WSMO) Feier et al. (2005). WSMO is a set of modelling elements for describing all aspects of semantic web services (Feier et al., 2005).

In this paper, we use the OWL-S infrastructure for specifying and accessing semantic web services. However, we believe that that the approaches we are developing for specifying and executing semantic web services are applicable to any target infrastructure.

3 Approach

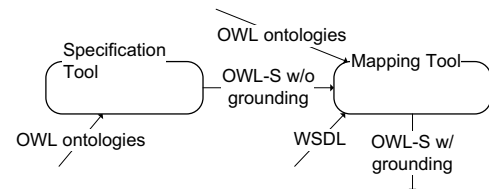
This section describes the approach that we are developing to support the construction of OWL-S specifications including the interactive technique for creating OWL-S groundings.

3.1 Overview

Figure 3 shows the process workflow for our approach. The framework uses a model-driven development approach for specifying, mapping and executing semantic web services. In this regard, the framework not only allows the specification of semantic web services but facilitates the mapping of constructs in those semantic descriptions to concrete service realisations. In this context, concrete realisations are typically traditional web services described using WSDL but have the potential to be other kinds of accessible services. These concrete service realisations can then be transformed into executable specifications for infrastructures such as BPEL (Andrews et al., 2003). In this framework, an architect is responsible for creating models using UML. The framework

manages the transformation process to OWL-S with very little additional effort on behalf of the developer. This benefits the developer in two ways. Firstly, the developer is able to focus on creating models instead of writing code or in the case of semantic web services creating a semantic specification. Secondly, the developer becomes more efficient because low-level details are abstracted away and the developer can focus more on the top-level structure and semantics. The developer can leverage existing skills in popular UML tools such as Poseidon (Gentleware, 2005) and Eclipse (Shavor et al., 2003).

Figure 3 Specification framework



The second stage of the framework involves a tool being developed that automates the process of mapping concepts in the OWL-S description to concepts in the WSDL file of a concrete service realisation (e.g. constructs a grounding). This tool uses the process model portion of the OWL-S description as well as a set of WSDL files for corresponding services to facilitate generation of the OWL-S grounding portion of the OWL-S description.

Once the grounding is created, the final step in the model-driven development process involves execution of the OWL-S specification. In order to leverage existing technologies, a tool is being developed, as part of the general framework, to automatically generate an executable application. Due to the nature of OWL-S groundings, these applications may have multiple concrete realisations from which completely separate executable specifications can be generated. As such, many different applications can be built from a single specification, thus forming a product family or product line of sorts.

3.2 Groundings

As shown in Figure 4, the OWL-S ontology (as represented by the *Service* class box) consists of three major parts. The *ServiceProfile* is a description of what the service does. In this sense, the service *presents* a particular process. The *ServiceModel* is a description of how the service works (e.g. the semantics and operational composition of the service). Within this context, a service may use either a single process or several atomic processes in ‘implementing’ the *ServiceProfile*. Finally, the *Grounding Model* is a description of how to access the web service. The grounding maps processes to specific realisations of the processes as implemented by web services.

Figure 5 shows an excerpt of a model for WSDL. From the perspective of a user of a web service, the structure showing the *PortType* and the contained *Operation* in the lower right hand corner of the diagram is most relevant. Specifically, the *PortType* provides access to the service. With respect to groundings, the *PortType* represents the target of a mapping from an OWL-S *Process*.

Figure 4 Structure of OWL-S Specifications

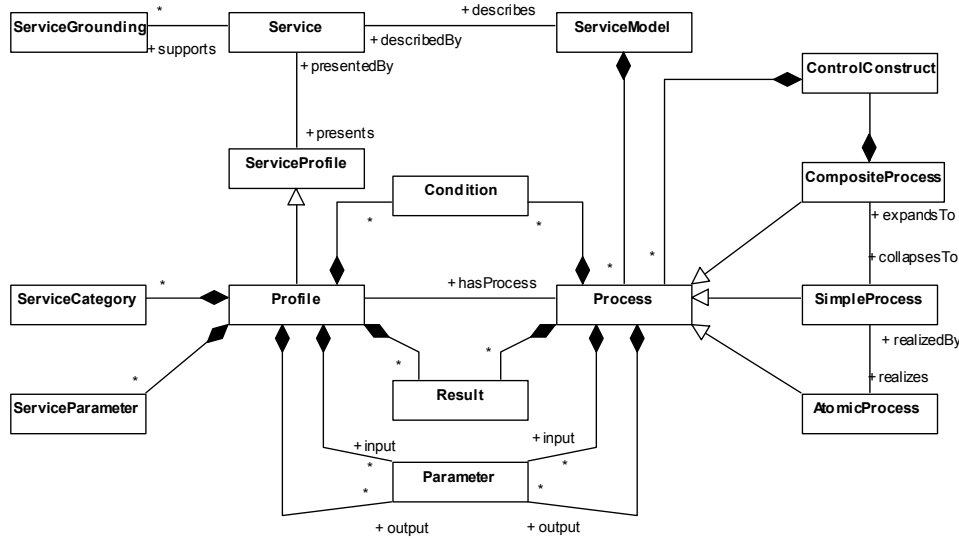


Figure 5 Excerpt of a model of WSDL

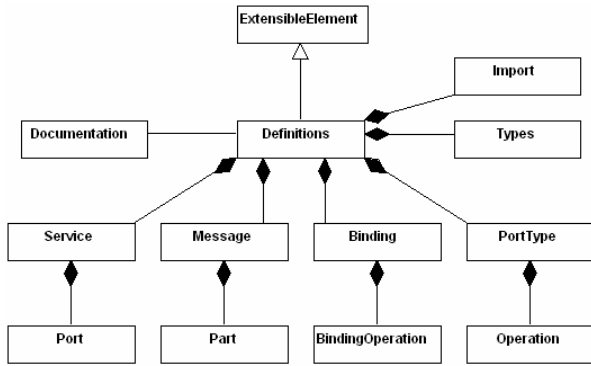


Figure 6 shows a UML diagram that captures the main idea behind groundings in an OWL-S specification. The diagram combines concepts from Figures 4 and 5. Specifically, a *ServiceGrounding* (shown on the top right) is made up of a number of *AtomicProcessGroundings*. These *AtomicProcessGroundings* contain mappings from *Process* types in an OWL-S specification to *PortTypes* in a WSDL specification. In the diagram, these mappings are represented with an association class called *ProcOpMap*. The *AtomicProcessGrounding* also contains mappings from parameters in an OWL-S specification to parameters in a WSDL specification and is captured with an association class (*ParaMap*) in a manner similar to that described above. In the case of parameters, the mappings can be realised by XSLT transformations as well as other complex translations from XML and XSD types to OWL types.

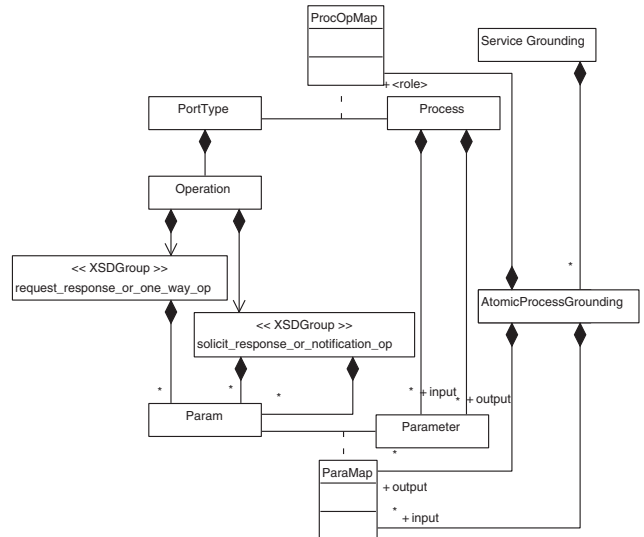
3.2.1 Mappings

There are two different perspectives that can be taken when creating OWL-S groundings. The first perspective comes from the viewpoint of software architects and knowledge architects that construct OWL-S specifications for semantic web services. This viewpoint can be considered to be a top-down view. From their perspective, the creation of an OWL-S specification is an activity of defining some abstract semantic service followed by the identification of services

that realise and implement the semantic service. From this perspective, it is desirable to identify groundings for all of the abstract services described in the OWL-S specification. Also, it is possible that a wide variety of services may satisfy the behaviour required by the semantic service and thus many groundings could be specified. In this context, it may be expected that the mapping activity will create a complete grounding, so that all of the behaviour required by the semantic service is identified and bound to the service. Accordingly, we offer the following definitions:

Definition 1: *Complete grounding.* Let s be an OWL-S specification for some semantic service, and $s.Op$ be the set of all processes defined in $s.profile$ (a profile of s) and $s.model$ (the service model of s). A grounding $s.grounding$ for s is said to be complete, denoted $complete(s, s.grounding)$, if for all processes o in the set $s.Op$, there is a well-defined mapping to a concrete realisation of o .

Figure 6 OWL-S to WSDL groundings



Based on the above definition, a grounding is a set of mappings from processes in an OWL-S specification to operations in one or more concrete realisations.

Definition 2: Well-defined mapping. Let s be an OWL-S specification for some semantic service and c be a web service. A mapping from a process o in $s.Op$ to an operation p in $c.Op$ is a relation associating the inputs and outputs of o to the inputs and outputs of p . A mapping from a process o in $s.Op$ to an operation p in $c.Op$ is well-defined if there is a bijective mapping from the inputs and outputs of o to the inputs and outputs of p .

A mapping is well-defined if all of the inputs and outputs of a given process in the set $s.Op$ described above are mapped to a corresponding input and output of a concrete realisation of the process. We currently handle only cases where mappings are well-defined. However, there are many cases where interfaces may be mismatched, thus requiring adaption and mediation.

Figure 7 graphically represents one kind of complete grounding. In left side of the diagram shows a Semantic Web Service (labelled ‘SWS’) while the right side of the diagram shows a conventional Web Service (‘WS’). Conceptually, among other things, both SWSs and conventional WSs specify a number of operations that are supported. In the diagram, the processes supported by the SWS are shown with a square while the conventional WS operations are shown with a triangle. The diagram itself shows that a grounding maps processes in SWS to operations in WS. In this case, the grounding maps all operations to the single service.

Figure 7 Complete Grounding to Single Service

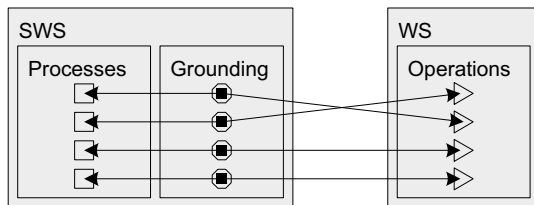
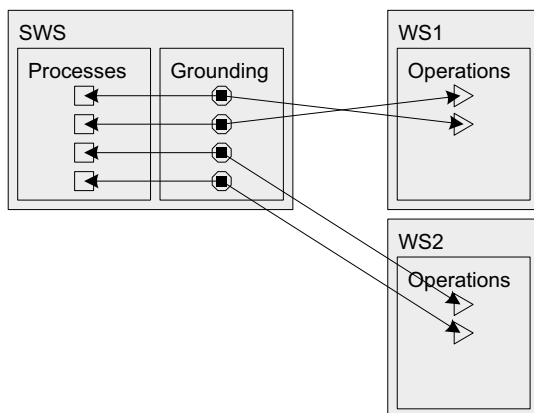


Figure 8 also shows a complete grounding but in this case, depicts a grounding that maps operations across several WSs rather than just a single service. This grounding is complete since all operations are mapped to some operation in a conventional WS.

Figure 8 Complete Grounding to Multiple Services



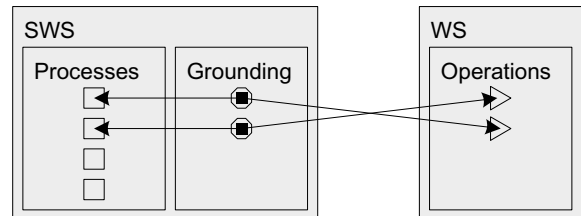
A second perspective for constructing groundings comes from the viewpoint of a service provider that may only be providing part of the behaviour expected in a semantic

service. This viewpoint can be considered a bottom-up view. For instance, consider the scenario of an e-commerce air travel ticketing service described as a semantic service in OWL-S. The service would include capabilities specific to the air travel domain such as searching for and reserving tickets for air travel. Other capabilities, however, may include general Business to Customer (B2C) operations such as shopping cart and credit card services. An OWL-S description of the e-commerce service could be specified with no specific groundings in mind so that organisations that do provide the fine-grained services could reuse the description, or rather ‘offer up’ their services as part of a collaborative effort to provide a composite service. In this scenario, it would be conceivable that a service provider for a specific component of the semantic service would generate a grounding just for the portion that they provide. Since they may not have knowledge to complete the grounding for what their collaborators provide, the grounding specification would be incomplete. In this context, we define an incomplete grounding as follows.

Definition 3: Incomplete grounding. Let s be an OWL-S specification for some semantic service, and $s.Op$ be the set of all processes defined in $s.profile$ (the profile of s) and $s.model$ (the service model of s). A grounding $s.grounding$ for s is said to be incomplete, denoted $complete(s, s.grounding)$, if there exists a process o in the set $s.Op$ for which there is no well-defined mapping to a concrete realisation of o .

Figure 9 depicts an incomplete grounding. As shown in the diagram, an incomplete grounding is simply a grounding that lacks defined mappings for at least one process in the semantic service.

Figure 9 Incomplete grounding



The definition for incomplete groundings gives rise to a requirement that in order for an OWL-S specification to be ‘executable’, amongst the many incomplete groundings that may be created for a service there must be a complete grounding made up of mappings in the incomplete groundings. The notion of an executable OWL-S specification is given below.

Definition 4: Executable OWL-S specification. Let s be an OWL-S specification for some semantic service and G with $|G| \geq 1$, be the set of all groundings for s , with a grounding $g \in G$ being a set of mappings from processes to operations. Specification s is executable if:

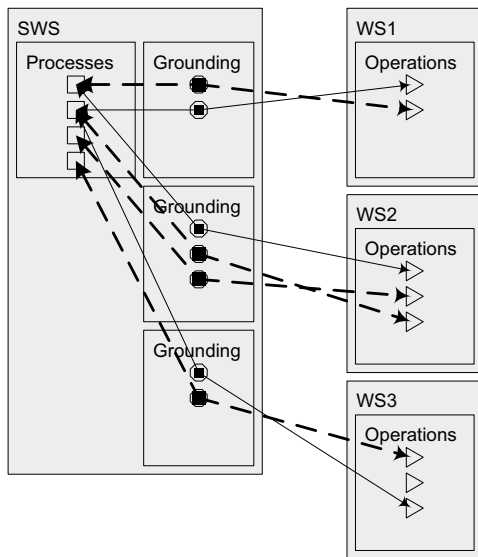
- 1 $\exists g \in G : complete(s, g)$ or
- 2 $\exists g' \notin G : \forall m \in g' (\exists g \in G : m \in g \wedge complete(s, g'))$.

Clause 1 states that if there is a complete grounding for a semantic service, then the OWL-S specification is executable.

The second clause says that if there is a grounding that can be induced (e.g. constructed by taking members from the groundings of the service) that is itself complete, then the specification is executable. The definition provides the basis for providing execution support in our overall framework. Specifically, it provides the mechanism identifying those specifications that can be executed and those that cannot.

Figure 10 shows a diagram depicting several groundings for a single semantic service. In this case, all groundings are incomplete. However, in amongst the union of the incomplete groundings, a complete grounding does exist, as indicated by the dashed lines.

Figure 10 Executable Grounding



The fact that many groundings can be created and induced for a given OWL-S specification has some desirable benefits. Specifically the existence of multiple groundings introduces a certain degree of variability into a semantic service in such a way that each grounding potentially provides slightly different behaviour. As such, special care must be taken to control how a given concrete service is selected when many groundings are present. In our approach, we currently rely on developers to make decisions regarding selection of a grounding. However, OWL-S as a language is flexible in that it allows a specifier to provide further annotations that can be used to differentiate between groundings. As such, decisions about selection of groundings could be made closer to runtime or at runtime. Such mechanisms, while out of scope of the work described in this paper, is among the many activities that we are interested in pursuing.

3.2.2 Multi-level multiplicity of mappings

One of the benefits of OWL-S is that it allows for general multiplicities when mapping from OWL-S services and processes to WSDL services. Specifically, OWL-S allows for *1 to many* mappings from specification to specification (e.g. a single OWL-S specification may be related to many WSDL specifications) and from OWL-S processes to WSDL operations (e.g. a single OWL-S process may be grounded to several WSDL operations and vice versa). From the standpoint of the specification level this should be no surprise

since many services can be involved in a collaboration to form one general business process. Likewise, at the process to operation level, there may be many implementations that realise a given atomic process.

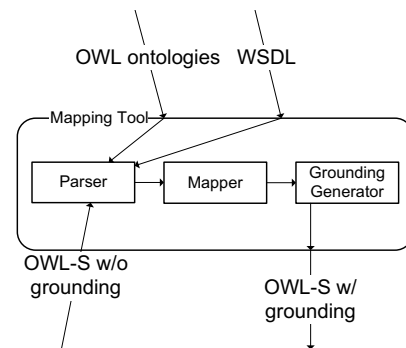
3.3 Tool support

The tool that we have developed to support the specification of groundings for semantic web services facilitates both top-down and bottom-up perspectives as described above by allowing a user to choose a view for describing mappings. In the case of the top-down perspective, a user is presented with a list of OWL-S processes that must be mapped to services in WSDL specifications. In the case of the bottom-up perspective, a user is presented with a list of WSDL operations that must be mapped to processes in an OWL-S specification.

3.3.1 Conceptual architecture

The conceptual architecture for the grounding tool is shown in Figure 11. The functionality of the system is essentially focused on three primary operations: parsing, interactive mapping and grounding synthesis. The inputs to the system vary based on the perspective taken by a user.

Figure 11 Conceptual architecture



When taking the top-down perspective, the inputs are an OWL-S specification and either a list of WSDL specifications or a list of search criteria for locating the WSDL specifications. The output for the tool in this perspective is typically a complete OWL-S grounding, although it can be incomplete if the mapping activity is either unfinished or the WSDL specifications do not cover all processes in the OWL-S specification.

When taking the bottom-up perspective, the inputs are an OWL-S specification and a single WSDL specification. In this case, the OWL-S specification can be optional as long as some mechanism exists to locate an appropriate OWL-S specification through some search mechanism. The output for the tool in this perspective is again an OWL-S grounding, but in this case, it is likely to be incomplete except in those case that the single WSDL file completely covers all of the processes found in the OWL-S specifications provided as input or through search.

The mapping activity, controlled by the mapper component in the diagram, is interactive in the sense that a user must identify the appropriate mappings between abstract OWL-S processes and concrete WSDL operations. We have

chosen to develop the tool to use an interactive approach rather than automatic one as an initial solution for generating groundings. We plan on developing a heuristic approach based on the use of search techniques that we have developed previously (Gannod and Bhatia, 2004). Currently the tool supports both input perspectives and allows a user to create mappings for atomic processes in an OWL-S specification and single WSDL specifications. The search capabilities are being built into the system in a future revision.

Once a grounding has been completed using the tool, it is combined with an appropriate OWL-S specification, depending on the input context. Since the groundings are syntactically correct, by construction, it is only necessary to check semantic correctness with respect to the original OWL-S specification. As a matter of practice, we have used the Protégé system (Gennari et al., 2002) to check both syntactic and semantic correctness of specifications. Specifically, we check that there are no violations in the generated grounding and that the entire OWL-S specification meets general constraints of the referenced ontologies. While Protégé is an editor, our intent has been to use aspects of the tool as a verifier. Clearly, a better strategy would be to use a verifier directly, which we are incorporating into our current investigations.

3.3.2 User interface design

Figure 12 shows a screenshot of part of the user interface for the mapping tool. This particular instance of the tool is shown from the bottom-up perspective where a WSDL specification containing several operations is being mapped to an OWL-S specification.

The OWL-S Mapping provides a simple and intuitive interface for users to generate a grounding instance from a WSDL and an OWL-S specification. We introduce to the features of the interface described in the context of the example in Section 4. The interface is divided into two symmetrical halves, the left pane and the right pane. In this example, the user selects to map a WSDL to an OWL-S

process. This perspective shows the WSDL in the left pane and the process file in the right pane.

The iterative steps to grounding generation are:

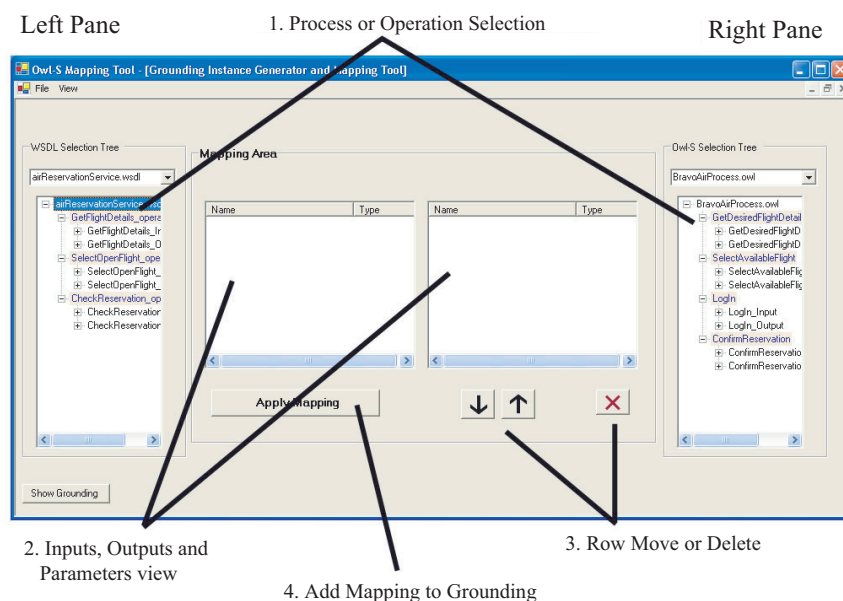
- 1 Select a WSDL operation name and process operation name
- 2 view properties and attributes of each
- 3 align the inputs and outputs into a complete mapping
- 4 click 'Apply Mapping' and repeat 1–4 as desired.

The selection area of the left pane shows each process in the WSDL, the inputs and outputs for that process, and the data types of the inputs and outputs. The selection area is designed as a tree view so the user can view the processes and associated attributes without restriction on navigation. Once the user finds a WSDL operation to include in this grounding instance, they click on the operation name to display the operation inputs, outputs and associated data types in the mapping area. Only one operation may be selected at a time. The currently selected operation name is shown above the left-pane mapping area.

The selection area of the right pane shows each concept in the OWL-S process file, the inputs and outputs for that concept and the data types of the inputs and outputs. This area, like the left pane, supports free navigation. A user clicks on the concept name to display the inputs, outputs and associated data types in the mapping area. Only one concept can be selected at a time. The currently selected concept name is shown above the right-pane mapping area. The mapping area is central on the interface of the tool. Here the user decides how the inputs and outputs of the selected WSDL operation correspond to the inputs and outputs of the selected Process. This mapping is realised when the user interactively arranges the input and output names so that the first row in the left pane maps to the first row in the right pane and so on with each row.

A complete grounding requires that all inputs and outputs in the WSDL be mapped to by an input and output in the Process. To facilitate complete groundings, the left-pane

Figure 12 Tool Interface



mapping area is not modifiable by the user. Conversely, the right-pane mapping area provides the user with buttons to modify row location or delete rows.

Once the user is satisfied with the mapping of inputs and outputs from the selected WSDL process to the selected Process concept, the user clicks the 'Apply Mapping' button. A grounding contains a definition for each WSDL operation that is mapped to each OWL-S. Groundings will most often contain several definitions. The user will add a definition to their grounding by repeating the iterative steps to generation grounding. The four steps in the grounding generation process are consistent with all mapping perspectives in the tool. These steps assume that the user chooses to map one WSDL file to one OWL-S process file. It is also necessary for the user to decide if the inputs and outputs are mapped in a way that makes semantic sense.

Currently the tool supports complete groundings to a single web service. Development of this tool is proceeding and planned improvements are underway. Future versions of the tool will support complete groundings to multiple web services.

3.4 Discussion

The intent of our approach is to provide a user with control over the grounding process, as is common with BPEL-based techniques. An alternative approach for grounding services is via the use of automated matchmaking, which is beyond the scope of the investigations described in this paper.

4 Example

In this section, we present an example demonstrating a usage scenario for the grounding tool. In this example, the user wants to generate a grounding for the OWL-S

PelicanSpellCheck service. This example demonstrates the fact that an OWL-S service can be grounded to WSDL services that on the surface may appear to be unrelated but after further examination result in a useful specification-operation pair. More specifically, the abstract OWL-S service specification describes a spell check process. We ground the process to the Google web service and the doSpellingSuggestion operation (Google SOAP Search API, 2006). Figure 13 shows a session corresponding to the PelicanSpellCheck example with the output of the session appearing in Figure 14.

The WSDL operation doSpellingSuggestion_operation is identified in the WSDL treeview. The user clicks on the operation name in order to view the operation inputs, outputs and parameters in the mapping area. The OWL-S process QuerySpellCheck is identified and selected from the OWL-S treeview. The inputs, outputs and parameters of QuerySpellCheck are shown in the mapping area.

In order to map the WSDL operation doSpellingSuggestion_operation with the OWL-S process QuerySpellCheck, the user needs to align the corresponding rows in the mapping area. In the left pane, the second row displays the WSDL parameter key. In the right pane, the second row displays the OWL-S parameter userID. These parameters are properly aligned, so no adjustments are required by the user in our example.

Suppose a case where the parameters are not properly aligned. Certain rows in the right pane would need to be moved so that the parameters in the left pane match in a way that makes semantic sense and creates a grounding that is correctly executable. In this scenario, the user clicks the up and down arrow buttons under the right pane until sufficient row alignment is complete. In certain scenarios a row that does not map may need to be deleted.

Moving on to the next input parameter in our example, the user must next map phrase, in the left pane, with queryPhrase,

Figure 13 User interface

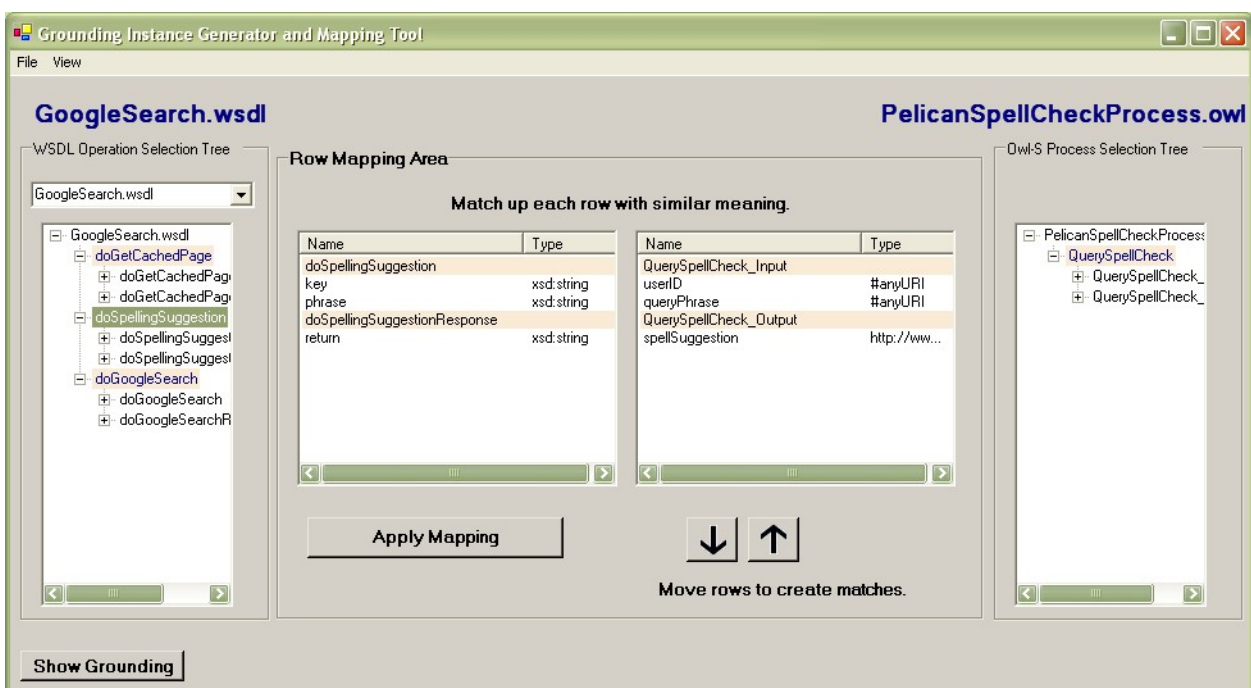
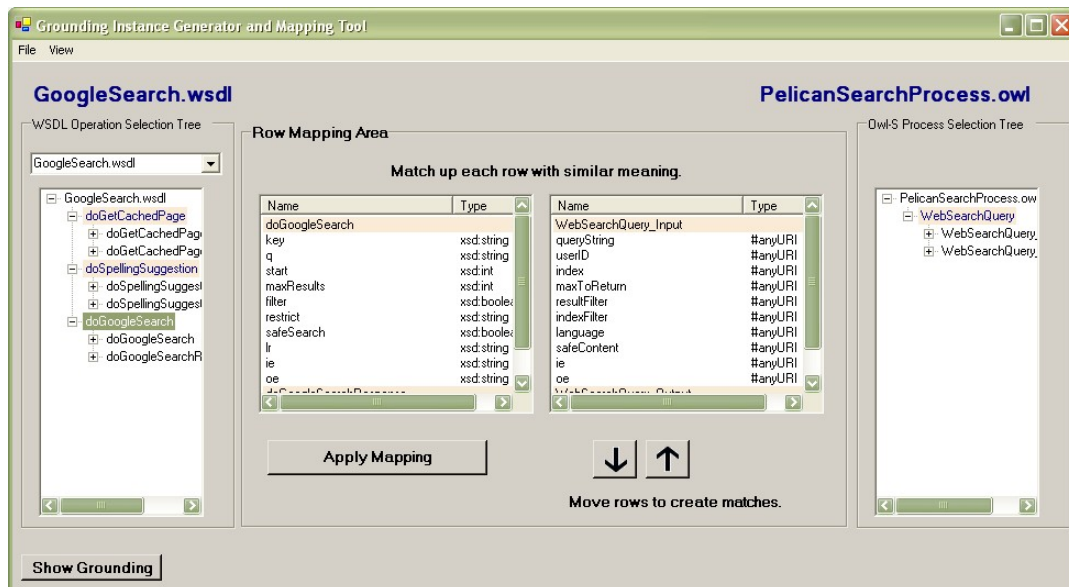


Figure 14 Excerpt of generated grounding

```

<grounding:WsdAtomicProcessGrounding
  rdf:ID="WsdGrounding_QuerySpellCheck">
  <grounding:wSDLDocument
    rdf:datatype=".../XMLSchema#anyURI">...GoogleSearch.wsd
  </grounding:wSDLDocument>
  <grounding:owlsProcess
    rdf:resource="...PelicanSpellCheckProcess.owl#QuerySpellCheck"/>
  <grounding:wSDLOperation rdf:resource="#doSpellingSuggestion_operation"/>
  <grounding:wSDLInputMessage
    rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
    ...GoogleSearch.wsd#doSpellingSuggestion_Input
  </grounding:wSDLInputMessage>
  <grounding:wSDLInput>
    <grounding:WsdInputMessageMap>
      <grounding:owlsParameter
        rdf:resource="...PelicanSpellCheckProcess.owl#userID"/>
      <grounding:wSDLMessagePart
        rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
        ...GoogleSearch.wsd#key
      </grounding:wSDLMessagePart>
    </grounding:WsdInputMessageMap>
  </grounding:wSDLInput>
  ...

```

Figure 15 Alternative bottom-up grounding for google

in the right pane. The user inspects the rows and verifies they align. These two parameters are properly mapped so, once again, no row manipulation is needed. Now that the inputs are verified to map properly, the user inspects the outputs. Return, in the left pane and spellSuggestion in the right pane are a correct map.

The rows are mapped as desired. The user clicks the Apply Mapping button to add these details to the grounding in generation. If the user wants to view the instance just generated, they click on the Show Grounding button.

To add more operation mappings to this grounding, the user iterates through another cycle of identifying operations, viewing the operation details in the mapping area and row matching via arrow buttons. The Apply Mapping button will add operations to the grounding. The grounding view shows the operation just generated and a grounding instance with each mapped operations defined in the hasAtomicProcessGrounding tags.

Once the user is satisfied that the grounding contains each operation they want mapped from GoogleSearch.wsd and PelicanSpellCheckProcess.owl, they save the grounding with

a specified name to a specified location. For the grounding to be executable, the user must update the URIs for the original WSDL and OWL-S files in the grounding as needed.

Figure 15 shows an alternative session for mapping the Google web service to an OWL-S process (in this case, a web search service). This view demonstrates the bottom-up nature of the grounding process. That is, it is possible for concrete services to map to several abstract OWL-S processes. Likewise, it is possible to map abstract OWL-S processes to several concrete services. As such, we could map the PelicanSearchProcess shown in Figure 15 to Google or to any other known search engine that is exposed as a web service.

5 Related work

The OWL-S Editor supports visual editing of OWL-S descriptions (Sciicluna et al., 2004). The tool provides a graphical user interface and supports generation of an OWL-S description from a WSDL specification in a

bottom-up fashion. This approach differs from ours in that it maps from WSDL to OWL-S but does not support top-down groundings. Our tool takes an OWL-S specification that have been generated either manually or using some automated tool (Timm and Gannod, 2005). Our tool interprets the OWL-S process and allows a user to manually define mappings between processes in the OWL-S model to operations in a WSDL file. This is particularly useful when creating multiple groundings from a set of existing services.

Paolucci et al. (2003) developed the WSDL2OWLS system for automatically generating a one-shot OWL-S specification for a given WSDL file (Paolucci et al., 2003). As such, their approach generates a complete grounding and incomplete profiles and process models. The approach works by translating a WSDL operation into a OWL-S atomic process and by translating XSD types into OWL-S concepts. Their original approach was developed for the DAML-S format (a precursor to OWL-S). From the perspective of the user, the WSDL2OWLS approach is a bottom-up technique. Our approach differs in that we use a top-down approach, preferring to develop high-level OWL-S specifications and then using the flexibility of OWL-S groundings to map the OWL-S services to any number of potential WSDL realisations of an OWL-S process.

A similar method of creating an OWL-S specification is taken by Shen et al. (2005). In their approach a BPEL4WS specification is translated into an OWL-S specification. As with the approach by Paolucci et al. this technique generates a complete grounding. However, they also generate a complete process model using the BPEL composition operations as guidance. As a result, only the profile is incomplete since no semantic information is used. The shortcoming of this approach is that it is limited to a single process instance and a single set of groundings. That is, it takes a bottom-up approach. Our approach takes a top-down approach and thus supports development of more abstract OWL-S specifications.

WSMO Studio for Eclipse is a WSMO-compliant editor for modelling semantic web services (Dimitrov et al., 2005). WSMO Studio supports roundtrip engineering of WSMO descriptions as well as composition through a choreography editor. WSMO Studio differs from our work it is based around WSMO as the ontology for describing services. We have, instead, chosen OWL-S for our ontology. Because our approach does not depend on a particular ontology language, we could develop transformations which convert UML models to WSMO based descriptions.

Finally, Balzer and Liebig (2004); Pantschenko et al. (2005) developed a semantic approach for mapping OWL-S parameter types to XML schema types through the use of an RDFS ontology for RDF mappings. This approach allows for a more seamless mapping from OWL-S to WSDL through the use of XSLT transformations. In our approach, we require that the user define how parameter types map. We find, as a result, that the approach by Balzer and Liebig has the potential to provide users with more assistance during the grounding process.

6 Conclusions and future work

Semantic web services are an evolutionary, if not revolutionary, next step in the advancement of web service technology. Though in its infancy, the upside of semantic web services far outweigh their downside. One of the primary challenges of using these technologies is one of population. That is, there are few semantic web services available for development and experimentation. In this paper, we describe a tool intended to overcome the population problem by facilitating the creation of groundings (e.g. mappings of service operations or processes found in OWL-S specifications to concrete and executable realisations of those operations as described by WSDL specifications). As a result, integrating conventional web services into an OWL-S-based infrastructure can be more simply achieved. The tool is meant to be part of an environment that currently includes an MDA-based tool for writing OWL-S specifications (minus groundings).

In our future investigations, we are planning on expanding the capabilities of the grounding tool to more completely support mapping activities including using the tool to assist with creation of operation mediators. Furthermore, we intend to more closely integrate this tool with the OWL-S specification tools that we are currently developing. Finally, we are developing an environment that will allow for the search, discovery, integration and execution of semantic web services. The tools we are developing currently facilitate specification of both atomic (Gannod et al., 2006) and composite services. Our immediate plans include support for execution.

Acknowledgement

Gerald C. Gannod supported in part by National Science Foundation Career grant No. CCR-0133956.

References

- Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Sheth, A. and Verma, K. (2005) 'Web service semantics – wsdl-s', Available at: <http://www.w3.org/Submission/WSDL-S/>.
- Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S. (2003) 'Specification: Business process execution language for web services version 1.1', Available at: <http://www-128.ibm.com/developerworks/library/ws-bpel/>.
- Balzer, S. and Liebig, T. (2004) 'Bridging the gap between abstract and concrete services: a semantic approach for grounding OWL-S', *Proceedings of the ISWC Workshop on Semantic Web Services*.
- Berners-Lee, T., Hendler, J. and Lassila, O. (2001) 'The semantic web', *Scientific American*, Vol. 284, pp.35–43.
- Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. (2001) 'Web service description language 1.1. W3C Note', Available at: <http://www.w3.org/TR/wsdl/>.
- Dimitrov, M., Simov, A. and Oguyarov, D. (2005) 'Wsmo studio – an integrated service environment for wsmo', *Proceedings of the Workshop on WSMO Implementations (WIW 2005)*.

- Feier, C., Roman, D., Polleres, A., Domingue, J., Stollberg, M. and Fensel, D. (2005) 'Towards intelligent web services: Web service modeling ontology (WSMO)', *Proceedings of the International Conference on Intelligent Computing (ICIC 2005)*.
- Gannod, G.C. and Bhatia, S. (2004) 'Facilitating automated search for web services', *Proceedings of the 2004 IEEE International Conference on Web Services*, pp.761–764.
- Gannod, G.C., Timm, J.T.E. and Brodie, R.J. (2006) 'Facilitating the specification of semantic web services using model-driven development', *International Journal of Web Services Research*, Vol. 3, No. 3, pp.61–81.
- Gennari, J., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubezy, M., Eriksson, H., Noy, N.F. and Tu, S.W. (2002) 'The evolution of protege: an environment for knowledge-based systems development', *International Journal of Human-Computer Studies*, Vol. 58, No. 1, pp.89–123.
- Gentleware (2005) 'Poseidon for uml', Available at: <http://www.gentleware.com/>.
- Google SOAP Search API (2006) Available at: <http://www.google.com/apis/>, Accessed on 14 September 2006.
- Kiko, K. and Atkinson, C. (2005) 'Integrating enterprise architecture representation languages', *Proceedings of the Workshop on Vocabularies, Ontologies, and Rules for The Enterprise*, pp.41–50.
- Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K. (2005) 'Bringing semantics to web services: the OWL-S approach', *Proceedings of SWSWPC 2004*, Vol. LNCS 3387, Springer-Verlag, pp.26–42.
- Miller, J. and Mukerji, J. (Eds) (2003) 'MDA guide version 1.0.1', *Technical Report omg/2003-06-01*, Object Management Group.
- OMG (2005) 'Ontology definition metamodel', *Technical Report OMG/RFP/ad/2003-03-40*, Object Management Group, 3rd revision.
- Paolucci, M., Srinivasan, N., Sycara, K. and Nishimura, T. (2003) 'Towards a semantic choreography of web services: from WSDL to DAML-S', *Proceedings of the International Conference on Web Services*, IEEE.
- Pantschenko, K., Noppens, O. and Liebig, T. (2005) 'Grounding web services semantically: why and how? W3C Workshop on Frameworks for Semantics in Web Services (W3C SWSF).
- Scicluna, J., Abela, C. and Montebello, M. (2004) 'Visual modeling of owl-s services', *Proceedings of the IADIS International Conference WWW/Internet*.
- Shavor, S., D'Anjou, J., Kehn, D., Fairbrother, S., Kellerman, J. and McCarthy, P. (2003) *The Java Developers Guide to Eclipse*, Addison-Wesley.
- Shen, J., Yang, Y., Zhu, C. and Wan, C. (2005) 'From BPEL4WS to OWL-S: integrating e-business process definitions', *Proceedings of the 3rd International Conference on Web Services (ICWS 2005)*, IEEE.
- Smith, M.K., Welty, C. and McGuinness, D.L. (2004) 'OWL web ontology language guide', *W3C Recommendation* Available at: <http://www.w3c.org/TR/owl-guide/>.
- Timm, J.T.E. and Gannod, G.C. (2005) 'A model-driven approach for specifying semantic web services', *Proceedings of the 3rd International Conference on Web Services (ICWS 2005)*, IEEE, pp.313–320.