

## Chapter 2

# Web service interface description languages

In this chapter<sup>1</sup> the web service description languages WSDL and OWL-S (formerly DAML-S) are investigated to determine how well they fulfill their role as IDLs for web services. Chapter 1 described three problems that inhibit dynamic ad hoc interaction; the problem of describing shared data, the problem of describing what services can do and the problem of interactively communicating to achieve goals. Section 1.2 discussed why WSDL and OWL-S cannot describe what services can do. In this chapter the focus is on how well these specifications allow the description of the data services require and provide.

The web services architecture is based on a “Publish, Find, Interact” model [81, 32]. This model is insufficient for web services to progress towards automated service to service interaction. A more complex model that includes description, advertisement, discovery, evaluation and selection, initiation of dialogue, negotiation, configuration, interaction, and management is required. The first step is description. Web service descriptions must contain sufficient information to allow each step to proceed automatically.

Web services are software therefore the requirements for ordinary software interface descriptions can be used as the basis for requirements for web service interface descriptions. The reason for doing this is to ensure that the description of web service interfaces is at least as comprehensive as the descriptions provided for other types of software.

The next section outlines the requirements for software interfaces described in [41]. In sections 2.2 and 2.3 WSDL and OWL-S are compared to these requirements. The results of the evaluation are presented in section 2.4.

---

<sup>1</sup>This discussion is based on [122]

## 2.1 Documenting interfaces

This section gives an overview of the interface documentation template presented in [41] and [9] with comments on how the sections apply in the web services context. These requirements have been refined over many years by the software engineering community. Each element is necessary to ensure that the developers implementing the interface, and clients using the interface, can fully understand what the interface requires, what it provides and its constraints.

1. **Interface identity:** The most common means is to give a name to the interface and version numbers if appropriate.

A unique identity for an interface is particularly important when many implementations of the same interface are expected or if different interfaces to the same service are provided for different classes of users. The identity can be used by service advertisements in catalogues and registries to indicate that the service complies with a particular interface.

2. **Resources provided:** These are the operations or methods provided in the interface.

The description of resources should be sufficient to aid the discovery, evaluation and selection of services that meet the users' needs.

Each resource needs to describe its:

- (a) **Syntax:** The signature including its name and the logical datatypes of arguments.
- (b) **Semantics:** A description of what happens when the resource is used i.e. what is visible to the user, and what are the restrictions on use of the resource. For example, the semantics describe the change in values of data that the user can access; the events signalled and messages sent by the resource; semantics detail how other resources will behave differently when this one is used; and the execution style, whether the operation is atomic, interruptible or suspendible.

The semantics of the operations is perhaps the most important part of the description in terms of enabling automated interaction. The semantics will be necessary in most phases from evaluation and selection to interaction.

- (c) **Usage restrictions:** Similar to pre-conditions, these state the assumptions about the environment that must be true, or describe the side effects of the operation. Exceptions should also be described here, detailing how errors are handled. For example, the number of retries or the meaning of returned status indicators.

3. **Locally defined datatypes:** This section describes how to declare variables, constants and literal values of the datatypes defined and used in the interface, and the operations, comparisons and conversions that can be performed on instances of those types.

Datatypes in the web context can be viewed as both traditional programming language constructs and XML documents. In the case of XML documents, a reference to the document schema or a document template can be provided to allow creation of required documents or to enable understanding of the documents supplied by resources.

4. **Error handling:** The errors that can be raised by the resources on the interface and error handling behavior.

This information will be necessary during the interaction phase to determine the cause of unexpected results.

5. **Variability provided by the interface:** Details of what configuration is possible, and range of allowable values for each configuration parameter should be provided. In addition, a description of how configuration affects the semantics of the interactions.

Web service users will be of many different types, operating in different contexts, each with different capabilities. Consequently, web service descriptions should provide the facility to describe what can be configured and how that configuration can be achieved. In addition to operation attributes, there are other aspects of service delivery that could be configurable such as, the interaction mechanism or the security and transaction management protocols.

6. **Quality attributes of the interface:** Quality attributes include such things as the level of performance and reliability that the interface provides.

Services may be selected based on their certified conformance to specific standards, or their ability to conform to quality of service requirements.

7. **What each interface element requires:** Either, specific named resources (described as above with syntax, semantics and restrictions), or other pre-conditions or assumptions about the environment.

The usage restrictions (or non-functional requirements) of the operations will be used in the evaluation and interaction phases to determine if the constraints can be satisfied by the user.

8. **Rationale:** The motivation for the design, the constraints, compromises and alternatives considered.

This is mainly for the benefit of developers implementing the interface.

9. **Usage guide:** the protocol of interaction or patterns of use for the entire interface.

This information will be used during the interaction phase to ensure the correct order of interaction. This information could also be used to describe how to initiate a dialogue with the service, how to negotiate with or configure the service, and how to manage the operation of the service. Alternatively, the information may be used to select services that provide compatible interaction mechanisms.

The interface requirements described here can be considered the minimum requirements for the interfaces of web services operating in the heterogeneous web environment. In the next section the template is used to determine how much of this information can be expressed using the current web-service specification languages WSDL and OWL-S.

## 2.2 WSDL evaluation

- Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 9 July 2002<sup>2</sup>

WSDL is the primary language for web service description and it has achieved a high degree of acceptance. It provides machine processability with XML syntax.

1. Identity: No specific identity attribute is provided by WSDL. When registered in a UDDI registry as part of a UDDI tModel<sup>3</sup>, a unique identity is assigned by the registry. The URI of the interface specification can be used as a unique identifier, however identical copies of the same interface with different URI's have to be treated as different resources.
2. Resources provided: A WSDL interface has a collection of operations (methods). Each operation has a set of input and output messages. Each message has one or more parts (parameters). Each message part has a name and a type. Although any type system can be used, XML Schema<sup>4</sup> datatypes are preferred. Messages are defined outside of an operation, so in theory they are reusable.

A WSDL operation also associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages as well as the sender and receiver.

- (a) Syntax: Operation signatures are provided in XML syntax according to the WSDL 1.1 XML Schema specification, for example:

---

<sup>2</sup>[www.w3.org/TR/2002/WD-wsd112-20020709/](http://www.w3.org/TR/2002/WD-wsd112-20020709/)

<sup>3</sup>[uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#\#\\\_Toc25137694](http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm#\#\_Toc25137694)

<sup>4</sup>[www.w3.org/XML/Schema](http://www.w3.org/XML/Schema)

```
<message name="aNameMessage">
  <part name="firstName" type="xsd:string">
</message>

<operation name="printName">
  <input message="aNameMessage"/>
  <output message="returnName"/>
</operation>
```

- (b) Semantics: WSDL makes no provision for semantics [22].
  - (c) Usage restrictions: Not supported.
3. Local data types: Defined using XML Schema complexTypes. No support for operations on the defined types.
  4. Error handling: Not supported.
  5. Variability: Not supported.
  6. Quality attributes: Not supported.
  7. Required resources: Input messages (as described above) are how required resources are described in WSDL. There is no support for semantics, usage restrictions or pre-conditions.
  8. Rationale: Not supported, although could be added as documentation.
  9. Usage guide: Not supported, although other specifications such as BPEL4WS<sup>5</sup> provide this kind of information by describing the order of invocation of WSDL operations.

## 2.3 OWL-S evaluation

- DAML-S 0.7 Draft Release October 2002<sup>6</sup>.

OWL-S (formerly DAML-S) is a web service description language based on the OWL ontology language. OWL-S markup is intended to facilitate the discovery, execution and interoperation of web services [111]. OWL-S provides three sub-ontologies, each providing a different view of the service. The Profile describes the organization providing the service, the names of the operations the service performs,

---

<sup>5</sup>[www-106.ibm.com/developerworks/webservices/library/ws-bpel/](http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/)

<sup>6</sup>[www.daml.org/services/daml-s/0.7/](http://www.daml.org/services/daml-s/0.7/)

and non-functional service characteristics. The Process Model describes how the service works in terms of its inputs, outputs, preconditions and effects. The Grounding describes how a service is used and provides a mapping from the Profile and Process specifications to specific concrete protocols and message formats such as WSDL.

This evaluation concentrates on the Service Profile, but describes elements from the Process model and Grounding when they provide more information. Although the various models provide a good separation of concerns they can be confusing when they appear to overlap. For example, the Profile describes parameters that have a name and an unspecified range (type). These Profile parameters make references to input and output parameters separately described with slightly different names in the Process model.

1. Identity: A Profile has a unique `serviceName` attribute.
2. Resources provided:
  - (a) Syntax: XML syntax is used with the vocabulary from the OWL and OWL-S specifications, for example:

```
<profileHierarchy:BookSelling
  rdf:ID="Profile_Full_Congo_BookBuying_Service">

  <!-- reference to the service specification -->
  <service:presentedBy
    rdf:resource="#congoService;#FullCongoBuyService"/>
  <profile:serviceName>
    Congo_BookBuying_Agent
  </profile:serviceName>

  <profile:input rdf:resource="#BookTitle"/>
  <profile:output rdf:resource="#ShippingOrder"/>

  <!-- specification of quality rating for profile -->
  <profile:qualityRating>
  <profile:QualityRating rdf:ID="Congo-Rating">
  <profile:ratingName>SomeRating</profile:ratingName>
  <profile:rating rdf:resource=
    "http://www.daml.ri.cmu.edu/ont/DAML-S/concepts.daml
      #GoodRating" />
  </profile:QualityRating>
  </profile:qualityRating>

  <!-- Preconditions and effects -->
  <profile:precondition rdf:resource="#AcctExists"/>
```

```
<profile:effect rdf:resource="#BuyEffectType"/>
</profileHierarchy:BookSelling>
```

- (b) Semantics: The Profile provides terms to name pre-conditions and effects. Processes can name conditions and conditional effects for parameters and processes.

Other aspects of semantics, such as the events signalled and how other resources will behave differently are not supported.

- (c) Usage restrictions: Some support, but the property *domainResource*, that describes resources necessary for the task to be executed, has been deprecated.

3. Local data types: Most OWL-S objects are defined locally as OWL classes. Operations on these classes are not supported.
4. Error handling: Not supported, although errors could be described in terms of *effects* as described in the Process specification.
5. Variability: Not supported.
6. Quality attributes: Various quality attributes can be defined for services, pre-defined attributes include *maxResponseTime* and *avgResponseTime*. The example above indicates how a quality rating would be described.
7. Required resources: Input parameters are usually described as instances of OWL classes. The OWL-S Grounding can provide a mapping to datatypes supported by WSDL.
8. Rationale: Not supported, although could be added as documentation.
9. Usage guide: The Process Control Model provides various control constructs including splits, joins, sequence etc. Composite processes can specify constraints on the ordering and conditional execution of sub-processes.

## 2.4 Summary

Table 2.1 gives a summary of the results of the evaluation. A “-” indicates no support, “+/-” indicates some support and “+” indicates reasonable support for the item.

Both specifications rate well on providing the syntax of operation signatures. However this is nearly all that WSDL does provide. This means that all the processes of discovery, selection etc. must be based on operation signatures, which is clearly inadequate for all but the most simple services. To progress beyond manual

Table 2.1: Summary of evaluation results

	<b>WSDL</b>	<b>OWL-S</b>
1 Identity	-	+
2 Resources provided		
2a Syntax	+	+
2b Semantics	-	+/-
2c Restrictions	-	+/-
3 Local Datatypes	+/-	+/-
4 Error handling	-	-
5 Variability	-	-
6 Quality	-	+
7 Resources required	+/-	+/-
8 Rationale	-	-
9 Usage guide	-	+

discovery, selection and pre-programmed interaction, WSDL needs to be expanded to include the other aspects of interface description, or other specifications (such as BPEL4WS) covering those aspects must be developed.

OWL-S provides a better interface description for semantics and the usage guide. The main problem areas are error handling and configuration. Errors are inevitable, and interfaces must declare what errors are possible and how they are handled, neither specification provides primitives for this kind of information. The lack of error handling, which is essential in production systems, may be a reflection on the immaturity of the specifications or a deliberate attempt to reduce their complexity. In either case, this issue must be dealt with before widespread commercial use of web services is possible.

Although there is no support in OWL-S for the definition of operations on locally defined data types, this is typical of ontology definitions. Ontologies focus on describing elements in terms of their relationships, rather than on how instances of the elements can be used.

## 2.5 Epilogue

In the 18 months since the initial evaluation of these specifications they have both been updated (at least once) with newer versions. In this section the table 2.1 is updated with notes on how the changes in these specifications alter the evaluation. Details of the most recent specifications are:

- Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language W3C Working Draft 26 March 2004<sup>7</sup> and
- OWL-S 1.0 Release October 2003<sup>8</sup>.

<sup>7</sup>[www.w3.org/TR/wsd120/](http://www.w3.org/TR/wsd120/)

<sup>8</sup>[www.daml.org/services/owl-s/1.0/](http://www.daml.org/services/owl-s/1.0/)

Table 2.2: Updated summary of evaluation results

	WSDL	OWL-S
1 Identity	+ <sup>1</sup>	+
2 Resources provided		
2a Syntax	+	+
2b Semantics	-	+/-
2c Restrictions	-	+
3 Local Datatypes	+/-	+/- <sup>4</sup>
4 Error handling	+/-	-
5 Variability	-	-
6 Quality	+/- <sup>2</sup>	+
7 Resources required	+/-	+/-
8 Rationale	-	-
9 Usage guide	+/- <sup>3</sup>	+

The changes in WSDL include use of the term “Interface” instead of PortType to represent a collection of operations and interfaces now have a unique name. The definition of an operation within an interface can include a boolean flag indicating whether it is “safe” i.e. it has no unexpected side-effects.

Operations declare which “message references” are bound to its attributes Input, Output, Infault and Outfault. This gives an indication of the direction and order of messages for this operation. A message reference is a reference to an XML element declaration specifying the message content. These element declarations appear to be defined in separate documents which are “included” in the WSDL service document, rather than being specified within the document itself as they used to be. Operations can also identify the message pattern that is appropriate for this operation. Message patterns are discussed in the context of service interaction in section 5.5.

The changes in the WSDL specification reflect a tightening of the overall architecture with a greater use of components with common properties. In my opinion the specification itself is becoming harder to read. It makes liberal use of navigation links for every reference to a concept, rather than effectively using links to provide a flow of relevant information.

There are minimal changes in OWL-S that impact upon the evaluation. The one of most interest is the increasing tendency to recognise XML Schema datatypes. This is in contrast to the previous version that opted for a looser definition by using the most general class OWL “Thing”.

The changes in OWL-S from version 0.7 to 1.0 represent a shifting of focus from the profile as a description of what a service can do, to use of the process model to describe what and how services are performed. The previous confusion over duplicated information in the profile and process model especially with regard to parameter definitions has been resolved by having the profile refer directly to the definition contained in the process model.

The consequence of the change in focus to the process model, is that an OWL-S service profile is intimately linked with its process model. Profiles cannot be used as implementation-independent declarations of functionality. Clients who discover service profiles, also need to access the process definition to find the missing details. As the process definition describes how the service works, the shift of focus appears contrary to the principles of abstraction and information hiding.

## 2.6 Discussion

The evaluation of the specifications reveals that web service interfaces created with WSDL and OWL-S provide much less information than is usually expected for software interfaces. It is not reasonable to expect services to operate in a complex environment like the web, and at the same time provide less information about them than is considered necessary for other software.

WSDL and OWL-S overlap in some areas, for example they both describe the syntax of operations well. Although some work is being done in OWL-S to align these two specifications [111], further effort should be directed at making web service interfaces more comprehensive.

In the heterogeneous environment that web services operate in there are many different cultural influences that affect the local choice of names for signature elements. As these elements are the only mechanism (apart from documentation that is not machine accessible) for determining the capability of a service they play an important role in the service description. There are ways WSDL and OWL-S could do more to promote cross cultural understanding of the terminology used in signature elements. A partial solution could be to mandate a naming convention. Other solutions include building-in the facility to provide alternative names, or the facility to “ground” names in a common source such as Wordnet. Such facilities are needed to cater for the diverse nature of the population that will need to discover and utilise services.

The results of the evaluation show that both WSDL and OWL-S need work before they can be considered as effective web service interface definition languages. In the light of this, the research reported in this thesis does not pursue the option of building additional layers on top of these specifications. The direction chosen here is to define conceptual level solutions to the problems identified. This leaves open the option of implementing the solutions within the context of these languages.

In the future when web services engage in automated ad hoc interaction, they will need to be able to describe and share information about their data, capabilities, interaction mechanisms. Web services will be self describing when they can do this

---

<sup>1</sup>A WSDL Interface now has a unique name

<sup>2</sup>A boolean “safe” attribute can be included in an operation definition

<sup>3</sup>A “pattern” attribute in the operation definition identifies a message exchange pattern

<sup>4</sup>There is an increasing use of XML Schema datatypes

as well as provide the core interface information necessary for any software interface discussed in this chapter.

