A Conceptual Architecture for Semantic Web Services

Chris Preist[1]

HP Labs, Filton Rd, Stoke Gifford, Bristol BS32 8QZ, UK
chris.preist@hp.com

Abstract. In this paper, we present an abstract conceptual architecture for semantic web services. We define requirements on the architecture by analyzing a set of case studies developed as part of the EU Semantic Web-enabled Web Services project. The architecture is developed as a refinement and extension of the W3C Web Services Architecture. We assess our architecture against the requirements, and provide an analysis of OWL-S.

1 Motivation

In this paper, we present an abstract conceptual architecture for semantic web services. The motivation behind this is (i) To provide a conceptual model to aid in the principled design and implementation of semantic web service applications; (ii) To allow the principled application of existing semantic web service ontologies such as OWL-S and WSMO; (iii) To contribute to the evolution of such languages, or the design of new ones, through the identification of appropriate features such a language needs; (iv) To motivate the development of appropriate infrastructure to support a variety of semantic web service applications. The main impetus behind this work has been provided by the European Union Semantic Web–enabled Web Services project (SWWS). This project aims to develop appropriate Semantic Web Services infrastructure to support four case-studies, inspired by the design principles proposed in the Web Services Modeling Framework [2]. Although these four case studies provide the primary source of requirements on our architecture, to encourage generality, we also consider a variety of other applications of SWS described in the literature.

We take the W3C Web Services Architecture (WSA) [14] as an important input. This provides a conceptual model for understanding web service technology and its relationship with other components. In particular, it adopts the view that the semantics of a service is provided by a contract (agreeing what the service does) between the service provider and service requestor. Such a contract can be explicit or implicit, and can be agreed in a variety of ways.

The potential role of contracts in supporting web service interactions is becoming increasingly recognised. The use of contracts in supporting interactions between agents in open systems has long been recognised by the agent community (e.g. [5]).

[1] The following people have contributed to this work through discussion and/or feedback: The HP Semantic Web Services team; the EU Semantic Web-enabled Web Services team, Dieter Fensel, Mark Burstein, Ian Dickinson, Terry Payne, Bijan Parsia and the ISWC reviewers.

Proposals have been made to use contracts in semantic web services [3]. The adoption of contracts as representing semantics in the WSA is further evidence of this trend.

Though the WSA adopts the contractual view, the implications of this are not explored as fully as is required if the semantics are to be made explicit. Specifically, it does not make a clear separation between the semantics of a service provider agent and the semantics of a particular performance of a service. Furthermore, it does not provide a model to show the activities involved in agreeing the 'semantics' of a particular service interaction (though it does informally discuss this process.) The conceptual architecture we present builds on the WSA to provide appropriate separation of concepts and additional constructs.

The WSA presents four different views on the architecture, focused around different core concepts; message, service, resource and policy. Our focus is primarily on the service-oriented view. In particular, we do not consider policy, security or reliability issues and how these are affected by the addition of explicit semantics. To do so requires additional analysis beyond the scope of this paper.

The paper is structured as follows. Firstly, we discuss the semantics of the word 'service', and consider the different ways it can be used. Next, we briefly describe the SWWS case studies, and refer to additional applications discussed in the literature. We then present a set of requirements on a semantic web services architecture, derived from analysis of this set of applications. Next, we briefly summarise the service oriented model of the WSA, and present our conceptual architecture as an extension of this. We assess the architecture against our requirements, present an analysis of OWL-S and conclude.

2 What Is a 'Service'?

The word 'service' can be used in several different ways. If a clear conceptual architecture is to be developed for semantic web services, we believe that it is necessary to make these different uses explicit.

1. 'Service' as Provision of Value in Some Domain
In the economy nowadays, a great variety of different services are traded; provision of stock quotes, shipment of crates, translation of documents, provision of a broadband connection etc. In each case, the customer receives something of value to them in return for a payment to the supplier. The service received can be described in terms of the domain of application. Such a description does not need to refer to the way in which the supplier and customer interact. This is what an economist or businessperson would refer to as a service.

On the internet, services can be provided which involve a payment, and so correspond closely to the above perspective. We can broaden this view of service without losing what is essential. A service can be provided without any payment, and the value the service provides need only be 'in the eyes of the requestor', not necessarily something with monetary value in the commercial world. For example, the provision of a specific piece of information from a database or the provision of a plan to achieve some goal. Hence, a service is *the provision of something of value, in the context of some domain of application, by one party to another.*

We also need to distinguish between a particular provision of value from the general capability to provide. We refer to the former as a *concrete service*, and the latter as an abstract service. Hence an *abstract service* is defined as *the capacity to perform something of value, in the context of some domain of application.*

2. 'Service' as a Software Entity Able to Provide Something of Value
This usage is common in the computer science and IT community. Papers often speak of sending messages to services, receiving results from services, executing services etc. While this is clearly appropriate in the context of web services and service oriented architectures, it can cause some confusion when mixing usage of this definition with the definitions above. For that reason, we believe that it is more accurate to speak of such an entity as being (part of) a service provider agent (SPA). We also use the term 'web service' to refer to a specific application with a WSDL/SOAP interface.

3. 'Service' as a Means of Interacting Online with a Service Provider
A common example of this usage is 'negotiation service'; A negotiation service provides the means to interact (negotiate) with a given actor. This usage is clearly different from usage 1, in that negotiation in itself does not provide something of value. However, the outcome of the negotiation may do so (and hence may be a service in the first sense.) For this reason, we refer to this as the provision of a negotiation 'protocol' or 'choreography'.[2]

All three of these aspects of service provision are essential; the 'value' in some domain, the interactions involved, and the implementation of the entity able to deliver what is needed. However, we believe that for conceptual reasons it is best to make clear distinctions between them.

3 Example Scenarios

In the interests of space, we present very brief overviews of five of the example scenarios used. Fuller descriptions are available in an extended version of this paper. Of these, scenarios 2, 3 and 5 are from case studies in the SWWS project.

Example A: Currency Conversion Quotation[3]
A simple service provider, providing a currency quotation for the conversion of a given amount of one currency into another currency, at a given time.

Example B: Design of a SPA to Proactively Manage Virtual ISP Problems
A system designer develops a complex service provider agent for managing faults. This requires the coordination of several internal applications, including a workforce management system (to timetable an engineer to fix the problem) and a customer relationship management system (to inform the customer). These applications have web service interfaces. The system designer develops a complex process out of these and 'wraps' the resulting system with a service provider agent front-end.

[2] A 'negotiation service' can exist in the first sense; this is a service where you 'subcontract' some negotiation problem to a service provider, and they negotiate on your behalf.

[3] By Terry Payne: http://www.daml.ecs.soton.ac.uk/services/SotonCurrencyConverter.html

Example C: Discovery, Selection, and Execution of a Customer Notification Service
A service requestor, OAS, needs to send a message to one of their customers. It does to by dynamically discovering a set of notification services, interacting with them to see if they can contact the specific person, selecting one based on functionality and cost, and interacting with it to send the message and provide payment.

Example D: 'Smart' Book Buying from Several Online Book Suppliers
A purchasing agent requiring several books locates several book sale SPAs, and interacts with them all simultaneously using their 'shopping cart' conversations to get quotes for purchase and delivery. It calculates the best SPA (or possibly set of SPAs) to use and 'checks out' an appropriate set of books, confirming the purchase. It monitors progress by using an 'order progress' conversation with the SPA.

Example E: Provision of a Logistics Supply Chain [1]
A company has a 'broken link' in a logistics supply chain and must rapidly locate a new shipping service between two ports. It makes a complex service request to a discovery service, and locates possible providers. It negotiates with them about the exact terms and conditions, and selects one. As the shipment takes place, it is accompanied by a complex exchange of messages between service provider and service requestor. This scenario requires two forms of 'service composition'. Firstly, the shipment service must 'slot in' with the other services in the logistics chain (arriving at the right time, etc). Secondly, the conversations with the service provider must be coordinated with conversations with other service providers in the chain.

4 Requirements Analysis

Based on the above case studies (and others) we identify requirements on a conceptual architecture (CA) for semantic web-enabled web services[4]:

1. The architecture should define the functionality of service discovery, but should not constrain the way in which it is enacted.
2. Discovery should be possible either using an explicit description of a required capability (E) or an implicit description through 'service provider type'.
3. Discovery identifies a set of SPAs which may meet the requestor's need. (C,D)
4. It should be possible to do discovery through informal browsing by a user (B).
5. Discovery should be symmetric; the functionality should not restrict a provider to advertise and a requestor to actively discover.
6. After the discovery process, there may (D) or may not (A) be a need for a contract agreement conversation to define the exact service to be provided.
7. The initiator should know prior to contacting them whether a conversation is necessary with the discovered agents to define the service.
8. A contract agreement conversation may involve negotiation of service parameters, selection of parameters from a set of possibilities (E), or a 'constructive' process such as the 'shopping trolley' metaphor (D).
9. A CA should enable negotiation to take place, but not place any requirements on either the structure of the contract agreement conversation required or on the negotiation strategies any participant may use

[4] The letters in brackets give the examples in section 2 which generate the requirement.

10. At the end of a contract agreement conversation, there will be an explicit contract which is identifiable as the 'semantics' (at the application level) of the agreed service. If there is no such conversation, then the discovery process should provide something identifiable as an explicit contract.
11. A service may be delivered directly as an output of an interaction with a Service Provider Agent (A), immediately as the effect of an interaction with a SPA (C), or as a later event or sequence of events after the defining interaction (E).
12. A post-agreement conversation may be directly linked to the execution/delivery of a service (E) or may simply monitor progress without directly affecting it (D).
13. A conversation about a service may be a simple request-respond pair(A), or may be a more complex interaction(D).
14. Conversations about a service may be initiated by the requestor(A), by the provider(E), or by either party (C)
15. A conversation may involve an asynchronous exchange of messages (E)
16. The semantics of a post-agreement conversation should be conceptually different from the domain-level semantics of the agreed service itself. However, there should be a means of linking the conversation with its domain level effect (E).
17. A service provider agent may in turn act as a service requestor agent to 'outsource' parts of the functionality it needs to deliver its service (B,C,D,E).
18. Some SPAs may achieve this outsourcing by executing processes of more simple elements created statically or dynamically using an internal planner (B); other SPAs may use very different reasoning methods (D).
19. It may be necessary to 'compose' service instances either to enable them connect in some way (E) or to provide a 'service bundle' that meets some need (D).
20. It may be necessary to 'compose'/coordinate conversations about services, to allow two services to interact appropriately (E).
21. Parts of the functionality required to support a conversation may themselves be treated as component services (D).

5 The Conceptual Architecture

In this section, we present a conceptual architecture (CA) which aims to meet the requirements listed in section 4. We now give a brief overview of the WSA service-oriented model [14] upon which our CA is based. The concept of a service is core, which is defined as an abstract resource representing a capability to perform some coherent set of tasks. The performance of a given task involves some exchange of messages between the requestor and provider agents. A choreography defines a pattern of possible message exchanges (i.e. conversations). The WSA associates a choreography with a task’s performance. The task may also have effects in the virtual or real world. Such effects may be publicly observable, but will often be private (such as a database update). The success or otherwise of a given task (from the perspective of one party) can be assessed by whether some goal state has been achieved. Such a goal state can be defined formally or informally.

As Sycara et al observe [10], tasks can be defined in three ways. They can be represented explicitly, using name labels with a well defined semantics in some ontology of tasks. They can be represented implicitly, using a language of preconditions and effects of a task. Or they can be defined using a combination of these two approaches. The WSA makes no commitment as to which approach is used.

A service has a description, which specifies its interface and messages in a machine-processable way. This may also include a description of the service's semantics. The semantics specifies (formally, informally or implicitly) the intended effect of using the service: specifically, the tasks that constitute it. The WSA views this semantics as 'a contract between the requestor entity and provider entity concerning the effects and requirements pertaining to the use of a service.' We adopt the majority of this model unchanged. The two changes we make are:

1. Distinction Between Abstract Service, Agreed Service, and Concrete Service
In the WSA, a service has a semantics which can be described, advertised and discovered by a service requestor. However, the service semantics is defined as the contract between the provider entity and requestor entity. This creates a problem: How can something which is defined as an agreement between the two entities be advertised prior to the two entities communicating? We believe this can be solved by making a distinction in the model between the *abstract service* a provider entity offers and the *agreed service* which appears within a contract, and assigning semantics to each. Additionally, (as outlined in section 2) we introduce the *concrete service* which is the performance of a specific service at a particular time. We give more complete definitions and examples of these concepts below. Similarly, we consider *abstract tasks*, *agreed tasks* and *concrete tasks*.

2. Distinction Between Service and Service Provider Agent
The WSA specifies that a Service Provider Agent realises a service. The word 'realises' (and the explanation in WSA) implies that the provider agent is an embodiment of the service. We believe that this confuses a service as the capability to provide something of value with the entity or software implementation which has the capability. For this reason, we say that a provider agent *provides* an abstract or agreed service, and *performs* a concrete service.

We now present our conceptual architecture(CA). Firstly, we present definitions of the concepts we introduce. Then we present and explain the architecture. For clarity, we separate it into three diagrams, representing the entities and relationships involved in discovery, service definition and service execution. We omit the entities and relationships which appear in the WSA if they remain unchanged in our model and are not relevant to the discussion of service semantics.

Firstly, we define the concept of a concrete service. This is the core 'building block' with which to define the semantics of more abstract descriptions of services.

Definition: A *Concrete Service* is an actual or possible performance of a set of tasks that represent a coherent functionality (and therefore deliver some value) within some domain of interest to its associated requestor and provider entities.

A concrete service: has a set of one or more concrete tasks; has a set of messages; has a service provider and service requestor; has one or more post-agreement choreographies associated with it; has a service provider interface and/or a service requestor interface which are used by the choreographies; may have a service description.

A concrete service is effectively a (possible or actual) specific and detailed set of tasks carried out at a given time, and the messages exchanged in association with this. The messages are exchanged between a specific provider and requestor, according to certain choreographies and using certain interfaces. A service provider, however, does not offer a specific concrete service. Rather, they are able to provide many different possible concrete services. Similarly, a requestor is rarely after a specific concrete service; they are interested in many possible services which meet their needs. For that reason, we define the concept of abstract service:

Definition: An *Abstract Service* is some set of concrete services. A concrete service is said to be a *realization* of an abstract service if it appears in this set.

Definition: An *Abstract Service Description* is some machine-processable description D which has, as its model, an abstract service C. (We write `Model`(D) = C)

An Abstract Service Description is some specification of a set of services. The format of this description is not specified as part of this architecture. There are several possibilities: (1) An extensional listing of concrete services. (2) A label representing the name of some abstract service in some ontology of services. (3) A template or formal description in some knowledge representation language.

Option one is the most conceptually straightforward, but in practice will be infeasible except in the simplest of cases. In option two, the semantics of the label is defined as the (possibly infinite) set of concrete services which correspond to that label within the ontology. This correspondence may be well-defined and explicit, with a service specification or contract template associated with the label. Often, however, the correspondence will be more informal. For example, a label 'bookselling' may have as its semantics the set containing any possible concrete services where books (and only books) are being sold. The third option specifies the set of concrete services by explicitly and formally defining an abstract service using some knowledge representation (KR) language such as OWL or F-logic. This may involve explicit descriptions of abstract tasks, possibly including parameters. The set-theoretic semantics of the KR language then defines the set of concrete services to be those which formally satisfy the concept description.

A resource representing an abstract service will have an abstract service description associated with it. In addition, it may be associated with: (i) a contract agreement choreography (ii) a service provider, and an associated service provider interface (iii) a service requestor, and an associated service requestor interface.

Whereas a concrete service is always associated with a specific requestor and provider, an abstract service may have one or both of these undefined. Similarly, an abstract task may be less completely defined than a concrete task is.

An abstract service, as we shall see below, can be used for advertising or requesting. However, it is not precise enough to define what a requestor and provider agree. For that we define the following;

Definition: An *Agreed Service* is an abstract service agreed between two parties Hence it is associated with a specific service provider and service requestor, together with their associated interfaces. It also has specific post-agreement choreographies associated with it. Its definition is such that: (i) any further decisions necessary to select a realization will occur as choices made during a post-agreement choreography. (ii) Any choice made by one party during the post-agreement choreography is such that the other party is indifferent to how that choice is made. The agreed service forms the core of the service contract between two parties, which defines the semantics of their interaction:

Definition: A *Service Contract* is an agreement between a service provider and requestor that the provider will supply an agreed service to the requestor. The contract: has a service provider; has a service requestor; has an agreed service description; has an association between the tasks involved in the agreed service description and choreographies and message interfaces which will be used to have conversations about the tasks; may have information about who is responsible and what compensation may occur in the case that a service is not delivered correctly; may have additional legal terms and conditions.

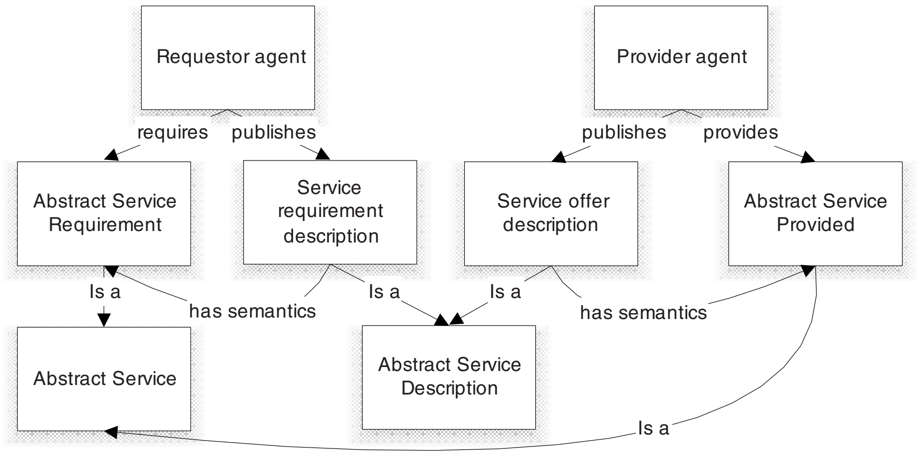
Requestor agent
Provider agent
requires
publishes
publishes
provides
Abstract Service Requirement
Service requirement description
Service offer description
Abstract Service Provided
Is a
has semantics
Is a
Is a
has semantics
Abstract Service
Abstract Service Description
Is a

Fig. 1. Discovery Model of Conceptual Architecture

The agreed service description will be (explicitly or implicitly) in terms of the tasks to be performed. It may include tasks to be performed by the requestor (e.g. payment transfer in the case of e-commerce applications.) The task/choreography association defines how the interaction between the parties can take place during delivery of the agreed service. This may be done by including the choreographies explicitly in the contract. This is appropriate where a choreography may vary and require agreement between two parties. Alternatively, it may be done by referencing a choreography definition elsewhere. This is appropriate when the service provider agent offers a specific choreography and is not able or willing to alter it, or provides a choice between a small number of choreographies, or the parties use some standard choreography defined elsewhere.

Fig. 1 illustrates the concepts involved in discovery. A requestor agent requires one of a class of abstract service; to this end, they publish a description of their requirements. Similarly, a provider agent is able to provide services within a certain class of abstract service. It publishes a description of the services it can provide.

Ideally, in each case, the description of the abstract service will correspond exactly to the actual abstract service which is required or offered. However, in practice, this may not be possible. For this reason, we define correctness and completeness of descriptions:

Definition: A description D of an abstract service C is said to be *correct* if any element in the model of D also appears in C. i.e. $\texttt{Model}(D) \subseteq C$.

Definition: A description D of an abstract service C is said to be *complete* if any element of C also appears in the model of D. i.e. $C \subseteq \texttt{Model}(D)$.

Often, it is possible to achieve completeness but not correctness in service descriptions. For example, a bookseller can state ‘I sell books’ in an advert, but cannot guarantee that every title is in stock. The role of the discovery process is to determine if a given service provider may be able to meet the needs of a given service requestor, and to inform one or both parties of this. As the WSA discusses, discovery can be achieved in several different ways, including the use of a discovery service. Note that we say that both the requestor and the provider publish a service description. Such a publication may be as an entry into some discovery service, as a query to such a service, or as a local advertisement or request in a peer-to-peer system [2]. Following requirement 5, we make no commitment as to which party advertises and which party queries. In general, we would anticipate that providers would advertise and requestors would query the advertisements; however, there are exceptions such as when a requestor issues a public Request For Quotes describing the kind of service it requires.

Assuming that descriptions of service requirements and service offers are complete but not correct, then the minimal functionality required of the discovery system is to determine if the two abstract service descriptions intersect [11]. If this is the case, an agreement between the two parties may be possible. Paolucci et. al. [8] describe a more sophisticated matchmaker which uses a hierarchy of different definitions of matching to determine which matches are more likely to be fruitful.

Following discovery, two parties need to determine if they can agree on a service instance which the provider will supply to the requestor. This is done through the contract agreement phase. In some cases, this phase will be unnecessary and the results from the discovery phase will be enough. This is the case if the following criteria are satisfied; (i) The published service offer description is correct (but not necessarily complete). (ii) The offer description provides a complete and explicit service description which includes post-agreement choreographies. (iii) Any aspect of the offer description which has variability (for example, a parameter which is not defined) is such that the provider is indifferent with regard to the choice.(iv) The delivery choreography (see below) is structured such that choices within this variability (such as a parameter instantiation) can be input through it.

If all four of these criteria hold[5], then the contract agreement phase is unnecessary. The service offer description defines a set of concrete services, and the provider doesn’t care which is selected. The delivery choreography allows the selection of a concrete service to be made by inputting parameter values chosen by the requestor. We call a service which has an abstract service description satisfying these criteria a **basic** service.

[5] This is phrased assuming the provider advertises and the requestor queries and selects. However, it is equally possible (though a lot rarer) that the requestor is indifferent, and the provider makes the selection using the requestor’s delivery choreography.

If the published abstract service description does not satisfy the criteria, but the service provider agent is able to give a abstract service description which does, then the contract agreement phase becomes a straightforward 'take-it-or-leave-it' process. The contract agreement choreography simply consists of the requestor accessing the description. It then goes on to execute the delivery choreography if it chooses to use this service provider, and does nothing otherwise.

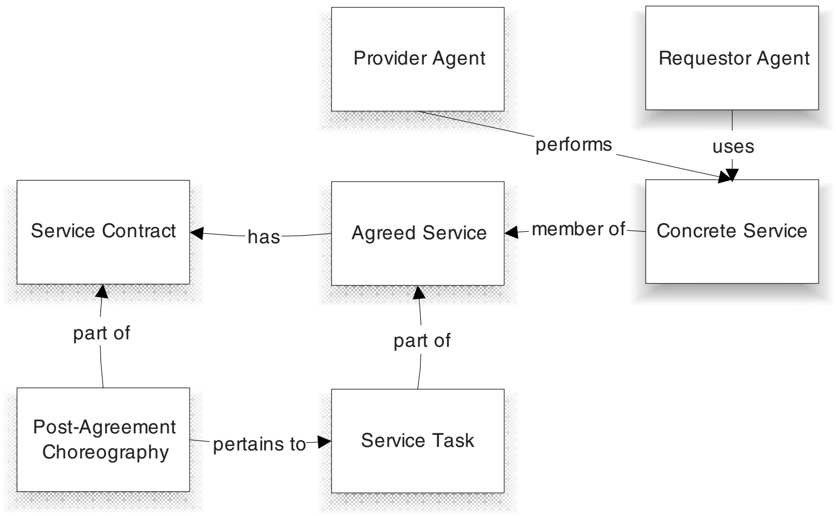

Fig. 2. Contract Agreement Model of Conceptual Architecture

In general, the contract agreement phase can be more complex than this. Fig. 2 illustrates the concepts it involves. At least one provider and at least one requestor agent participate in a choreography to agree a contract between them. The choreography involves discussing some abstract service, and may explicitly involve the manipulation of a description of it. If the choreography terminates successfully, it results in a service contract agreed between two parties; this contract contains a description of an agreed service which specifies the concrete service to be delivered. The agreed service will be a specialization of the abstract service found during discovery.

The decision making within the contract agreement phase has three aspects: ***Partner selection***: Often, one or both parties can choose which partner will deliver/receive a service. *Service selection*: One party, usually the requestor, may need to select and specify certain parameters of a service instance which weren't specified during discovery. *Negotiation*: Often, when a service specification is fully defined, other contract parameters (e.g. price) will be fixed. However, in some cases, they may be open to negotiation.

Any given contract agreement phase does not necessarily involve all three of these. Furthermore, they can take place in any order, and may be interleaved.

Definition: A *contract agreement choreography* is a choreography between at least one service requestor and at least one service provider which, on successful termination, results in the agreement of a service contract between two of the parties.

Definition: A *service contract* is an agreement between a service provider and requestor that the provider will supply a specific service instance to the requestor. A service contract has (i) a service provider, (ii) a service requestor and (iii) a service instance with associated service description and resource identifier.

A contract agreement choreography may explicitly involve service selection and/or negotiation. Service selection protocols can take different forms: A service provider may offer a list of all contracts they are willing to offer, and the requestor simply selects one. A requestor may iteratively build a definition of the service they wish to receive using the 'shopping trolley' metaphor, and are given a price quote for it. A requestor may refine a service template through dialog with the provider about different options available. Example negotiation protocols include propose/counter-propose (where each party proposes a contract, and the other party either accepts it or proposes an alternative) and constraint relaxation (where parties relax constraints on a contract template until both sets are satisfiable [12]). Many other possible approaches to negotiation exist ([4], [13]).

The association of a contract agreement choreography to a given set of provider and requestor agents must take place following discovery. The simplest way to do this is for the advertising agent to associate a (pointer to) a contract agreement choreography description with their advert, or to return a contract agreement choreography when they are first queried directly.

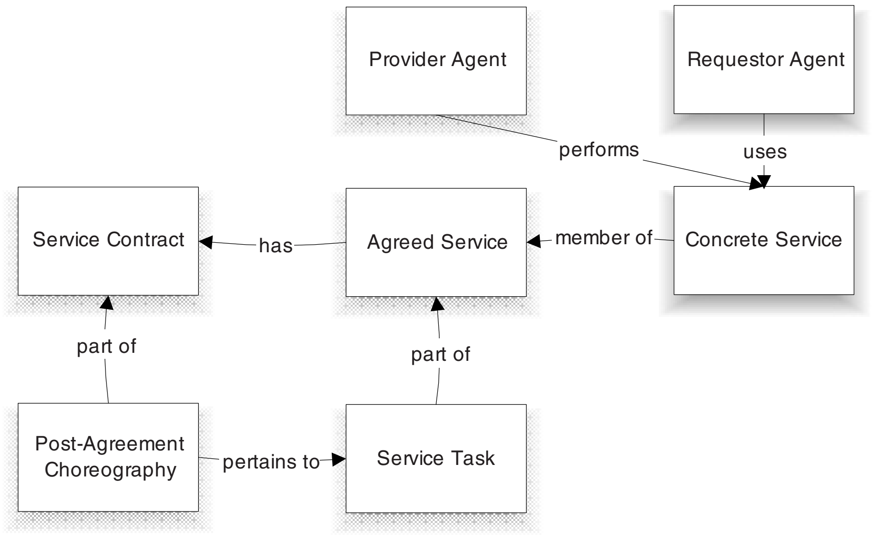

Fig. 3. Service Delivery Model of Conceptual Architecture

When a contract agreement choreography has terminated successfully, then service delivery can take place. This may commence immediately on agreement of the contract, or may be initiated later by a subsequent choreography. Fig. 3 illustrates the concepts involved in the service delivery phase. A concrete service is supplied by the provider agent to the requestor agent. This concrete service must be a member of the agreed service (which, recall, is a set of concrete services). The agreed service has an associated service contract which contains a description of the service agreed together with a mapping from the agreed tasks within the service to choreographies about these tasks. To distinguish these choreographies from the contract agreement choreographies, we refer to them as post agreement choreographies:

Definition: A post-agreement choreography is a protocol for a conversation between a service requestor and service provider about a contract which has been previously agreed, or one or more tasks within such a contract.

There are at least three different kinds of post-agreement choreography. A *delivery choreography* governs conversations which directly initiate and/or control the execution of one or more tasks in a contract. A *monitoring choreography* allows the requestor to monitor the task's progress. A *cancellation/renegotiation choreography* allows the termination or alteration of a service within a contract in certain circumstances.

This completes the Semantic Web-enabled Web Services conceptual model (CA). We now assess the architecture against the requirements of section 4.

6 Assessment of the Conceptual Architecture

Requirements 1, 2, 3 and 5 are straightforwardly satisfied by the structure of the conceptual architecture applied to the discovery phase as discussed above. Requirement 4, that discovery can be performed by informal browsing as well as matching, is not satisfied immediately by the architecture. However, a browser can easily be implemented within the conceptual framework described above; a set of service offer descriptions can be filtered, ordered and presented to a user according to certain parameters specified by a user.

To meet requirement 6, the architecture allows but does not require a contract agreement choreography between the parties. Furthermore, we characterise services which do not need such a choreography, and define them as basic services. While this characterisation is not technically part of the architecture, a discovery service could use it to determine if a contract agreement choreography will be necessary, satisfying requirement 7. Requirements 8, 9 and 10 are straightforwardly satisfied by the structure of the conceptual architecture as applied to the contract agreement phase as discussed above. In the case of basic services, an explicit contract is provided by the service offer description which is offered as take-it-or-leave-it and so needs no further refinement.

The architecture makes a clear separation between the definition of a service, interactions defining the service, and interactions involved in the delivery of the service. In particular, the delivery of a service can take place independently of its definition, satisfying requirement 11 and part of 16. The requirement that there be a means of linking a conversation with its effect is enabled but not satisfied by the architecture; use of a language which allows this is necessary. The other requirements on service delivery choreographies (12, 13, 14, 15) are straightforwardly satisfied by the structures defined above.

The conceptual architecture, like the WSA, allows 'outsourcing' of functionality (17) as agents can act simultaneously both as service providers and service requestors, possibly of several different services. We place no restrictions on the internal reasoning used to determine how this is done. One option is to use a planner to determine which service requests to make (18).

By separating out the definition of an agreed service from the choreographies to define and deliver the service, the architecture allows different kinds of 'composition' to take place. In particular, it enables parties to reason over possible service definitions at the domain level during the contract agreement phase to ensure coordination between the different services agreed(19). It also enables parties to reason about, and coordinate, choreographies associated with the agreement and deployment of different services (20). However, both of these are merely enabled by the architecture; reasoning strategies to do these must be embodied within specific requestor/provider agents.

A third form of what could loosely be called 'composition' is the application of some service to enable a choreography between provider and requestor of some other service (21). For example, a 'shopping trolley' choreography to define a book purchase may be enabled by component services, with functionality such as 'book search' or 'put in trolley'. The domain of application of these component services is 'shopping trolley choreography' as opposed to 'book delivery', the domain of the 'higher level' service being defined. However, within their domain, they can be considered as services with associated contractual definitions, and can be located and reasoned over by an agent which brings about the choreography between requestor and provider of the book delivery service. Such an arrangement, as opposed to fixed choreography definitions, would enable more flexible interactions between the two parties. It is conceivable to imagine a hierarchy of services nested in this way; though in the end it would bottom-out to fixed choreographies or basic services.

7 Analysis of OWL-S with Respect to the Conceptual Architecture

OWL-S ([7],[9]) (formerly DAML-S [6]) is an ontology for use in providing semantic markup for web services, developed by a coalition of researchers as part of the DARPA Agent Markup Language program. In this section, we compare the conceptual model implicit in OWL-S with our conceptual architecture.

The OWL-S ontology defines a service in terms of three top-level classes; the profile, the service model and the grounding. The profile characterises what a service requires and provides, and is used primarily in the discovery phase. The service model describes how a service works, primarily in terms of a process model. This defines a service's functionality as the execution of a (potentially complex) process of component web services. The grounding defines how a component web service can be interacted with, primarily through WSDL/SOAP. These three classes roughly correspond to our requirement for a service definition at the domain level, a description of how to interact with a service provider, and an implementation of that interaction. However, there are some differences which mean that OWL-S would need to be adapted and augmented if it were applied within our architecture.

In OWL-S, the profile is used almost exclusively as an advertisement/request. Hence it focuses on the description and categorization of an abstract service, but does not consider what is necessary to refine it to define an agreed service and to use this as a specification of what interactions accomplish. To do this, the service model (either within or outside the profile) would need to be augmented to provide complex structured descriptions of abstract services, and the ability to constrain and instantiate service parameters. The profile does not make a clear distinction between what a service does and the way in which one interacts with a service provider to receive the service. In particular, it is not clear that inputs/outputs should appear in the domain-level description of an agreed service. (For example, the domain level agreed service description may state that 'provider pays £10 by credit card'; the associated choreography will require a credit card number as its input. Hence the domain level definition and the choreography to deliver it are kept separate.)

The process model in OWL-S appears to serve two separate functions. Firstly, it can be used to define the exchange of messages with a service provider about a service; secondly, it can be used to define how a service provider implements the functionality of a service as a process of component web services. Both of these are valid uses, but the conceptual architecture requires that they be separated out.

The required exchange of messages about a service is defined by one or more post agreement choreographies in our conceptual architecture. The process model could be used to represent this. Note that the architecture can have several choreographies associated with a service, each performing different functions. Hence a given service may need several process models, and links between the process models and the domain-level service description to define what each does.

The definition of a service's functionality in terms of a process over component web services is a particular method of delivering a service. The CA requires that such a method should not be proscribed and that a provider using such a method should not be required to make public their method. For this reason, using the process model in this way would not be compliant. A separate structure defining the internal process should be used, and the publication of it would be optional. (Such a separation is being adopted in the nascent WSMO [15])

8 Conclusions and Future Work

We have presented a conceptual architecture developed out of a requirements analysis of a set of case studies in the SWWS project and example applications from the literature. The architecture makes a clear separation between the domain-level definition of a service to be provided, the message exchanges which bring about the service, and the internal logic which provides the appropriate functionality. Inevitably, the architecture is at an abstract level; however, it does shed light on the required characteristics of service ontologies and infrastructure services to realise it. We are currently working on an extension of the CA to include sub-communities of service agents which share a problem-solving method (in particular, planning). We also hope to expand it to cover mediation more effectively.

The conceptual architecture is being used to define, design and implement a general technical architecture for Semantic Web Services able to demonstrate the four case studies within the SWWS program. It is also being used as the underlying conceptual model in the development of WSMO-full [15] and has been submitted for consideration by the architecture subcommittee of the Semantic Web Services Initiative.

References

1. Esplugas-Cuadrado, J., Preist, C. and Williams, S.: Integration of B2B Logistics using Semantic Web Services. To appear, Proc. Artificial Intelligence: Methodology, Systems, and Applications, 11th International Conference, (2004)
2. Fensel, D. and Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce: Research and Applications, 1 (2002) 113-117
3. Grosof, B. and Poon, T.: SweetDeal: Representing Agent Contracts with Exceptions using Semantic Web Rules, Ontologies and Process Descriptions. To appear, International Journal of Electronic Commerce (2004)
4. He, M., Jennings, N.R. and Leung, H: On Agent Mediated Electronic Commerce. IEEE Transactions on Knowledge and Data Engineering 15(4) (2003) 985-1003
5. Jennings, N.R., Faratin, P.,Johnson,M.J., O'Brien,P. and Wiegand, M.E.: Using Intelligent Agents to Manage Business Processes. Proceedings of the First Int. Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (1996) 345-360
6. McIlraith, S. and Martin, D.: Bringing Semantics to Web Services. IEEE Intelligent Systems, 18(1) (2003) 90-93
7. OWL-S 1.0 Release. http://www.daml.org/services/owl-s/1.0/
8. Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K: Semantic Matching of Web Service Capabilities. Proc. International Semantic Web Conference (2002) 333-347
9. Paolucci, M. and Sycara, K.: Autonomous Semantic Web Services. IEEE Internet Computing, (September 2003) 34-41
10. Sycara, K., Paolucci, M., Ankolekar, A. and Srinivasan, N.: Automated Discovery, Interaction and Composition of Web Services. Journal of Web Semantics 1(1), Elsevier (2003)
11. Trastour, D., Bartolini, C. and Gonzalez-Castillo,J.: A Semantic Web Approach to Service Description for Matchmaking of Services. In Proceedings of the Semantic Web Working Symposium, Stanford, CA, USA, July 30 - August 1, 2001
12. Trastour, D., Bartolini, C. and Preist, C.: Semantic Web Support for the B2B E-commerce pre-contractual lifecycle. Computer Networks 42(5) (August 2003) 661-673
13. Vulkan, N.: The Economics of E-Commerce. Princetown University Press, Princetown, New Jersey (2003)
14. W3C. Web Services Architecture. W3C Working Group Note, 11 February 2004. http://www.w3.org/TR/ws-arch/
15. Web Services Modelling Ontology: http://www.wsmo.org/