Provider-Composer Negotiations for Semantic Robustness in Service Compositions*

Nikolay Mehandjiev, Freddy Lécué, and Usman Wajid

The University of Manchester
Booth Street East, Manchester, UK
(firstname.lastname)@manchester.ac.uk

Abstract. Research in automating service composition is rarely concerned with service providers, apart from work in quality guarantees and contracts. This perspective is arguably valid for comparatively static and cheap web services, which do not warrant continuous involvement of their providers in the process of service procurement and use by service consumers. However, opportunities for optimisation and fine-tuning of compositions are thus missed. We have created an approach which uses automated agent-based negotiation between service composer and service providers to address the issue of semantic robustness in large-scale service compositions by preventing cases where the wrong type of data is passed on from one service to the next. Starting from a service composition template which is not semantically robust, we allow the selection of semantically robust combinations of actual services. The approach is characterised with a linear complexity and also allows service providers to tune their services to the requirements of service compositions which may be lucrative business opportunities.

Keywords: service composition, semantic services, semantic robustness, autonomic agents, negotiation, template-based composition.

1 Introduction

Services are perceived as ubiquitous software-based units which can be procured by their consumers at the point of need to deliver certain functionality [1]. When a consumer desires functionality which cannot be provided by a single existing service, we can either develop a new service "from scratch", or attempt to compose one using existing services. Service composition is thus a valuable activity, and automating it has become a popular topic for service researchers, which have created a bewildering variety of approaches and methods.

One such approach [2,3] uses formalised knowledge about generic problem-solving approaches to break-up the desired functionality into a set of simpler units, called tasks. These tasks are interlinked into a service composition template, and suitable services are then sought for each task. If a number of services are found, one is selected aiming to optimise the composition according to certain criteria. For example, [4] shows how we can select a set of services which fit in terms of input and output data types.

* Foundation Project: Supported by European Commission VII Framework IP Project Soa4All.

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 205–220, 2009.
© Springer-Verlag Berlin Heidelberg 2009

We say that such composition is *semantically robust* if we cannot have the wrong type of data passed on from one service to the next. Reasoning about services in general and semantic robustness in particular is greatly facilitated by tagging services with formal semantic descriptions of their functionality, inputs, outputs, pre-conditions, etc. These services are then known as *Semantic web services* [5].

Their formal semantic descriptions are based on Description Logics (DL) [6], such as OWL-S [7], WSMO [8] or SA-WSDL [9] (through annotations). These are in general specialisations of semantic tagging languages such as the Web Ontology Language (OWL) [10]. The latter are used to provide semantic annotations for general web resources, including documents and media streams, thus creating the Semantic Web [11].

The problem we address in this paper is how to achieve semantic robustness of the service composition if the composition template we use is not semantically robust itself. This may occur for a number of reasons, for example when we modify a semantically robust template to include specialised functionality, or if the service composition template is created manually by people. For brevity we will use "robust" instead of "semantically robust" in the remainder of this paper. A non-robust template will specify the desired services in a way which permits the selection of incompatible services, i.e., one service generates output which does not conform to the specifications for the input of the follow-up service. For example, a voice transcription service may handle English and German, whilst the follow-up grammar checking service may be specialised in English only. The latter will thus fail if it is given a German text as an input. The failure could be at the level of functionality, in that it will detect all phrases as grammatically incorrect, but it may also raise exception regarding bad input since the input string will now have extra characters from the German language (e.g., ä, ü or ö) which are not expected by an English grammar checker software.

There are several approaches to resolving this issue. For example, we can convert the composition template to one which is robust by narrowing down the specifications of the respective service outputs, and only then we start searching for candidate services. This may exclude many valid compositions and produce sub-optimal results, especially if some inputs are "over specified" unnecessarily. For example, there may be many multi-lingual grammar checker services available to complement our example bilingual voice transcription service and result in a robust composition.

Such combinations will be detected and used by an alternative approach which analyses every combination from the two sets of candidate services in the hope of finding robust matches. This will work for small numbers of candidate services, but its complexity is exponential and thus not applicable for large-scale compositions. In addition, the resulting combinations may deviate from the template prescriptions significantly, impeding the straightforward substitution of a failing service in the future.

In this paper we address the issue of robustness by approaching it from a multi-agent systems perspective which is rarely used in web service research. We involve the service composer and the corresponding providers of the candidate services in a negotiation process aiming to result in a robust composition optimised according to their perspectives. The composer and providers can be represented by autonomous software entities, called agents, which are pre-programmed to negotiate according to the business

interests of the organisations they represent, and to reason over the semantic specifications of requirements and candidate services.

To drive the negotiation dialogue, we use a formal model of semantic robustness [12] described in Section 2. The model allows us to analyse each data link between tasks in our template, and for those links which are not robust, calculate precise semantic specifications of the *extra description* necessary for these links to become robust.

The negotiation process is guided by negotiation protocols, involving the service composer agent and the agents providing candidate services for every two tasks linked by a non-robust link. An informal outline of the protocols, together with overall approach proposed here, is described in Section 3, whilst the formal details of sub-protocols and negotiation strategies are specified in Section 4.

The combination of agent-based negotiation with semantic reasoning results in an innovative solution to the problem of robust service composition. Section 5 demonstrates how the approach can be used in a specific case study, delivering results with greater flexibility than the approaches based on centralised reasoning. Section 6 compares the approach with related work in the area, and Section 7 concludes this paper.

2 Preliminaries

This section describes in further detail the overall ideas of template-based service composition, and proceeds to define the formal model of their semantic robustness.

2.1 Template-Based Service Composition

An intuitive view to service composition would see it as an activity which aims to satisfy the need for a (non-existing) service by bringing together existing ones. For example, if we need a letter dictation service we can bring together a voice transcription service, a grammar checker service, a letter layout service and a printing service.

This integration activity can be done manually, yet automating it makes it more in tune with the vision of composing services at the point of need [1]. Automating can be done using program synthesis and AI planning techniques [13], employing reasoning over the pre- and post-conditions of available services, trying to create a plan of putting them together to jointly achieve the aim of the target composite service. This approach starts "from scratch" every time, yet significant performance improvements may be offered by reusing composition results as a template for new compositions, or creating such a template through the use of domain-specific knowledge about how the problem addressed by the sought service would decompose into sub-problems [14].

We follow this template-based composition, and focus on the stage of *template instantiation* [2,3], where we need to allocate a specific service for each the generic "service slots" in the template. From the perspective of template instantiation, we use the specification of each task $T_{i,1\leq i\leq n}$ to procure a set of candidate services $s_{j,1\leq j\leq m}$ for this task, and to select one of these services to instantiate the task. The precise manner in which we propose to implement both the procurement and selection activities so that we achieve semantically robust composition even in the cases where the template itself is not semantically robust, will be described in Section 3.

2.2 Formal Semantic Model

Using tasks specifications of inputs, outputs, pre- and post-conditions of templates, we should be able to infer additional dependencies between tasks, for example we can infer data flow dependencies between tasks using their input and output specifications.

In the following we present such dependencies as *semantic links* [15] between services. Then we define the concept of their *robustness* and finally we describe *semantic-link-based web service composition*.

Semantic Links. Since input and output parameters of semantic web services are specified using concepts from a common ontology[1] or Terminology $\mathcal{T}$ (an example of such is given in Figure 2), retrieving links between output parameters $Out\_s_i \in \mathcal{T}$ of services s_i and input parameters $In\_s_j \in \mathcal{T}$ of other services s_j could be achieved by using some DL reasoner such as Fact++[2] [16]. Such a link, also known as semantic link [15] $sl_{i,j}$ (Figure 1) between two functional parameters of s_i and s_j is formalized as

$$\langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j \rangle \tag{1}$$

Thereby s_i and s_j are partially linked according to a matching function $Sim_{\mathcal{T}}$. This function expresses which matching type is employed to chain services. The range of $Sim_{\mathcal{T}}$ is reduced to the four well known matching type introduced by [17] and the extra type Intersection [18]:

- **Exact.** If the output parameter Out_s_i of s_i and the input parameter In_s_j of s_j are equivalent; formally, $\mathcal{T} \models Out\_s_i \equiv In\_s_j$.
- **PlugIn.** If Out_s_i is sub-concept of In_s_j; formally, $\mathcal{T} \models Out\_s_i \sqsubseteq In\_s_j$.
- **Subsume.** If Out_s_i is super-concept of In_s_j; formally, $\mathcal{T} \models In\_s_j \sqsubseteq Out\_s_i$.
- **Intersection.** If the intersection of Out_s_i and In_s_j is satisfiable; formally, $\mathcal{T} \not\models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.
- **Disjoint.** If Out_s_i and In_s_j are incompatible i.e., $\mathcal{T} \models Out\_s_i \sqcap In\_s_j \sqsubseteq \bot$.

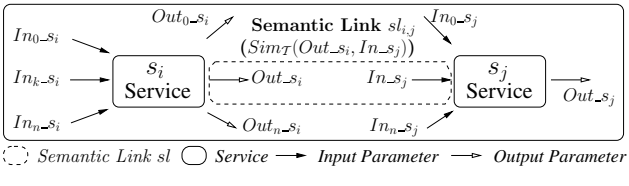

Fig. 1. A Semantic Link $sl_{i,j}$

In the same way as semantic links $sl_{i,j}$ between web services s_i and s_j, we define abstract semantic links $sl^A_{i,j}$ between tasks T_i and T_j. In the following we extend the definition of semantic link by introducing its concrete form (Definition 1).

[1] Distributed ontologies are not considered here but are largely independent of the problem addressed in this work.

[2] http://owl.man.ac.uk/factplusplus/

Definition 1 ***(Concrete Semantic Link)***
A concrete semantic link $sl_{i,j}^{\alpha,\beta}$ is a concretization of its abstract form $sl_{i,j}^{A}$ if and only if s_α and s_β can respectively concretize tasks T_i and T_j.

Robust Semantic Link. The matching function $Sim_{\mathcal{T}}$ of semantic links enables, at design time, determining the degree of semantic compatibility among independently defined web service descriptions, from the strongly compatible *Exact* through *PlugIn*, *Subsume* and *Intersection* to the strongly incompatible *Disjoint*. However, as emphasized by [19], the matching types Intersection and Subsume need some refinements to be usable for semantic-links-based web service composition.

Example 1 ***(Semantic Link & Subsume Matching Type)***
Suppose T_1 and T_2 are two tasks such that the output parameter `NetworkConnection` *of T_1 is semantically linked to the input parameter* `SlowNetworkConnection` *of T_2. According to the example ontology in Figure 2, this abstract semantic link $sl_{1,2}^{A}$ is valued by a Subsume matching type since $NetworkConnection \sqsupset SlowNetworkCon\text{-}nection$. It is obvious that such an abstract semantic link should not be directly applied in a service composition since the* `NetworkConnection` *is not specific enough to be used by the input parameter* `SlowNetworkConnection`*, which may cause data-based exception during execution. Indeed the output parameter* `NetworkConnection` *requires further restrictions to ensure a data-robust composition of T_1 and T_2.*

A semantic link valued by the Intersection matching type requires a comparable refinement. In this direction, [19] defined a robust semantic link and their composition.

Definition 2 ***(Robust Semantic link)***
A semantic link $\langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j \rangle$ is robust iff the matching type between Out_s_i and In_s_j is either Exact or PlugIn.

$$
\begin{array}{l}
NetworkConnection \equiv \forall netPro.Provider \sqcap \forall netSpeed.Speed \\
VeryRestrictedNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.AdslVeryRestricted \\
LimitedNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.AdslLimited \\
SlowNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.Adsl1M \\
FastNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.AdslMax \\
AdslVeryRestricted \equiv Speed \sqcap < 1mBytes \\
AdslLimited \equiv Speed \sqcap \geq 0.5\,mBytes \sqcap \leq 1.5mBytes \\
Adsl1M \equiv Speed \sqcap \geq 1mBytes \\
AdslMax \equiv Speed \sqcap \geq 8mBytes \\
AdslSuperMax \equiv Speed \sqcap \geq 16mBytes \\
Address \sqsubset \top,\ IPAddress \equiv Address \sqcap \forall protocol.IP \\
VoIPId \equiv Address \sqcap \forall network.Telecom
\end{array}
$$

Fig. 2. Sample of an $\mathcal{ALN}$ Terminology $\mathcal{T}$

A possible way to replace an Intersection-, or Subsume-type link $\langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j\rangle$ with its robust form consists of computing the information (as DL-based description) contained in the input parameter In_s_j and not in the output parameter Out_s_i. This information is then used as an additional restriction on the Out_s_i data type when a suitable web service is procured. We say that adding this latter restriction "transforms" the non-robust semantic link in its robust form. To do this, we apply initial ideas of [12], which adapt a non standard inference matching type i.e., the *Abduction* operation [20] (Definition 3) for comparing $\mathcal{ALN}$ DL-based descriptions.

Definition 3 ***(Concept Abduction)***
Let $\mathcal{L}$ be a DL, C, D be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. A Concept Abduction Problem (CAP), denoted as $\langle \mathcal{L}, C, D, \mathcal{T}\rangle$ aims at finding Extra Description, as a the most general concept $H_{C,D} \in \mathcal{L}$ such that $\mathcal{T} \models C \sqcap H_{C,D} \sqsubseteq D$.

According to Definition 3, a compact representation of "*difference*" $H_{Out\_s_i, In\_s_j}$ (henceforth H_{s_i,s_j}) between DL-based descriptions Out_s_i and In_s_j of a semantic link $sl_{i,j}$ can be computed. Such a description H_{s_i,s_j} can be formally defined by $\mathcal{T} \models Out\_s_j \sqcap H_{s_i,s_j} \sqsubseteq In\_s_i$ as a solution of the *Concept Abduction* problem $\langle \mathcal{L}, Out\_s_i, In\_s_j, \mathcal{T}\rangle$. In other words the *Extra Description* H_{s_i,s_j} refers to information required by In_s_j but not provided by Out_s_i to ensure a correct data flow between web services s_i and s_j.

In the same way robustness can be computed in template-based composition e.g., in case of non robust abstract semantic links $sl^A_{i,j}$ between tasks T_i and T_j. In the following H_{T_i,T_j} will refer to *Extra Description* between T_i and T_j in template-based composition (with non robust abstract semantic links).

Example 2 ***(Robustness and Extra Description)***
Suppose the abstract semantic link $sl^A_{1,2}$ in Example 1. The additional restriction which has to be provided to the `NetworkConnection` *if this output is to be used by the input parameter* `SlowNetworkConnection` *is referred by the* Extra Description *H_{T_1,T_2} of the Concept Abduction Problem $\langle \mathcal{L}, NetworkConnection, SlowNetwork\text{-}Connection, \mathcal{T}\rangle$ i.e., $\forall netSpeed.Adsl1M$ (see Figure 2).*

In other words, we can turn non-robust semantic links into robust ones by retrieving their *Extra Description*.

Semantic Link Composition Model. Here, we aggregate the concept of web service composition and semantic link in a same model. Therefore the process model of web service composition and its semantic links is specified by a directed graph which has the web service specifications s_i as its nodes, and the semantic links $sl_{i,j}$ (data dependencies) as its edges. In the same way a template-based composition, pre-computed for instance by *template-based* and *parametric-design-based* approaches [2,3], has the tasks specifications T_i as its nodes, and abstract semantic links $sl^A_{i,j}$ as its edges.

Given a template-based composition and an approach to compute robust semantic links (Definition 3), we address the issue of automating robustness in web service composition by using agent-based negotiation.

3 Negotiating Robust Interfaces with Candidate Service Providers

In an ideal template-based service composition, all semantic links between tasks (service placeholders) would be semantically robust. In practice this may not be the case, for example because the template has been created manually, or a generic template such as "object loan" has been modified with a domain-specific task such as checking credit record (for high-value objects such as expensive cars).

In such cases, we propose an agent-based approach to achieve robust instantiation of the non-robust template, which uses the formal model of semantic composition defined in the previous section. The approach is based on the following:

1. Every service provider and the service composer are represented by software agents.
2. The service composer agent "advertises" the service composition template on a shared notice board. It also calculates which semantic links in the template are not robust.
3. Service provider agents monitor the notice board. When they see requirements (task specifications) which one of their services can satisfy, they would "bid" for their service to instantiate the task.
4. Once the bids have been placed, the service composer agent initiates a three-phase negotiation protocol for each non-robust abstract link $sl^A_{i,j}$ in the template. The protocol involves the providers of services s_i and s_j which are candidates to instantiate the tasks T_i and T_j, respectively. The protocol should select services which provide robust instantiation of the semantic link.
5. The service composer agent can now instantiate the remaining tasks in the template by choosing the most appropriate service (in term of its semantic links with other services) for each such task.

In the remainder of this section, we will detail the suitability criteria used by service provider agents, followed by details of the three-phase negotiation protocol detailed in Step 4 above.

3.1 Service Suitability

Here we consider that a task T of a template can be instantiated by a service s if and only if the following conditions are true:

1. The service s achieves the same goal as T, assuming an ontology of goals [8].
2. The pre-conditions of s are implied by the pre–conditions of T.
3. The post-conditions of s imply the post-conditions of T.
4. The matching type between the input specification In_T of T and the input specification In_s of s i.e., $Sim_{\mathcal{T}}(In\_T, In\_s)$ is PlugIn.
5. The matching type between the output specification Out_s of s and the output specification Out_T of T i.e., $Sim_{\mathcal{T}}(Out\_s, Out\_T)$ is PlugIn.

Conditions (1) to (3) above ensure the candidate service s has the desired effect of the target task T, whilst conditions (4) and (5) ensure the semantic (functional) fit between

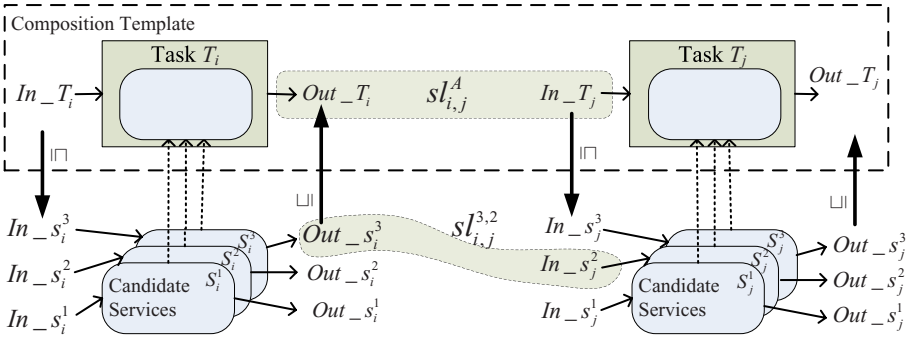

Fig. 3. Links between Tasks and Their Candidate Services

the candidate service and the target task. Condition (4) ensures that all the data which can be passed onto T can be processed by s. Condition (5) ensures that the output of s fits within the output specifications of T. Fig. 3 demonstrates the nature of the semantic fit between tasks and their candidate services.

Example 3 ***(Tasks and Suitable Services)***
We illustrate our approach by considering two different tasks T_1 and T_2 such that:

- `AdslEligibility` *task T_1, starting from a* `PhoneNum`*, a* `ZipCode` *and an* `Email` *address, returns the* `NetworkConnection` *of a desired geographical zone;*
- `VoiceOverIP` *task T_2, starting from a* `PhoneNum` *and a* `SlowNetworkConnection`*, returns the* `VoIPId` *of the ADSL line a Telecom operator needs to install the line;*

On the one hand T_1 can be concretized by three services:

- *s_1^1, s_1^2 and s_1^3, that, starting from a* `PhoneNum`*, a* `ZipCode` *and an* `Email` *address, returns respectively a* `SlowNetworkConnection`*,* `VeryRestrictedNetworkConnection` *and* `LimitedNetworkConnection` *of the desired geographical zone;*

On the other hand T_2 can be concretized by two services:

- *s_2^1 and s_2^2, that, respectively starting from a* `NetworkConnection` *and* `SlowNetworkConnection`*, returns the* `VoIPId` *of the ADSL line a Telecom operator needs to install the line;*

Note that s_1^1, s_1^2 and s_1^3 are suitable services for achieving task T_1 since they fulfil conditions (1), (2), (3) and (4). In the same way s_2^1 and s_2^2 are suitable services for T_2. In the rest of the paper we will focus on concretizing tasks by adequate services to achieve semantically robust links.

3.2 Negotiation Protocol

The service composer agent has identified all non-robust abstract semantic links $sl^A_{i,j}$ between tasks T_i and T_j in the composition template. The composer agent has also calculated H_{T_i,T_j} for each non-robust link. Once all the bids to instantiate the tasks involved in these links with services have come through (say an announced deadline for bidding has passed), the service composer will initiate a 3-phase negotiation protocol with the service providers for each *non-robust link* as follows.

Phase 1: In this phase all agents operate on the basis that they may achieve robust composition "for free" (i.e. without the use of extra services or modifying the behaviour of the ones proposed), using differences in specifications between a task and its candidate services (c.f. Section 3.1). We start by contacting all providers of services s_i for task T_i (on the left of Figure 3) sending them H_{T_i,T_j}. They compare it with their output specification as detailed in Section 4.1 to check if their (more specific) outputs turn $sl_{i,j}$ into a robust link. This is feasible since for each such output we have $Out\_s_i \sqsubseteq Out\_T_i$. If one or more service providers confirm this is indeed the case, the composer agent can terminate the protocol and, using the same selection criteria as the ones applied for a robust link, select one of them, and also any service provider for T_j. The actual selection criteria for choosing an instantiation could be based on a number of configurable parameters such as price, quality guarantees, etc. and will be application-specific. Alternatively, some service provider agents can provide their precise output specifications, if they have satisfiable intersection with the request (see Section 4).

In the second step of this phase, the service composer circulates the counter-offers (Out_s_i) to all providers of services s_j for task T_j (on the right of Figure 3), to check if their In_s_j (which subsume the input specification of their task In_T_j), covers at least one of the counter-offers in a PlugIn type of link and thus make the link robust. If $Sim_{\mathcal{T}}(Out\_s_i, In\_s_j)$ is of PlugIn type for at least one pair of candidate services, the respective service provider for s_j will respond to the service composer, and the protocol will terminate with success. Otherwise each service provider will return a counter-offer which is the extra description required for this concrete semantic link $sl_{i,j}$ i.e., $\mathcal{T} \models Out\_s_i \sqcap H_{s_i,s_j} \sqsubseteq In\_s_j$.

Phase 2: In this phase all agents operate on the basis that additional services will be required to make the link robust, and that the service consumer will have to pay additional usage fees for these extra services. They attempt to find just a single additional service per non-robust link, and to avoid having to modify or create services. This phase starts with the service composer contacting the service providers s_i ("on the left"), with either the specific "paired" counter-offers H_{s_i,s_j} generated from Phase 1, or, where the agent has not secured such a "paired" offer, with the general H_{T_i,T_j}.

The service providers then try to find the extra service (possibly in coalition with another service provider), which provides the missing semantic information and thus can narrow Out_s_i and thus convert $Sim_{\mathcal{T}}(Out\_s_i, In\_s_j)$ into the robust PlugIn type. If they succeed, they will respond with the cost of using this extra service. In this case the service composer agent will terminate the protocol, and select one of the services with such offers, using its usual criteria. Therefore, in that specific case, the agent does

not actively modify the service behaviour, but rather finds new services that support this extra description to ensure compliance to the restriction at run-time.

If no such offer is received, the service composer agent will contact all service providers "on the right", asking them to consider finding extra services which can act in parallel with their offerings and extend their specification of In_s_j to a degree where there is a PlugIn relationship with any of the Out_s_i. If no such offers are found, the negotiation proceeds to Phase 3.

Phase 3: At this phase all agents operate on the basis that some degree of service adaptation and/or development is necessary to achieve robustness of the specific semantic link, and the expectations of monetary values are thus also increased. Again we use the pairs of offers and counter-offers derived in the previous phases, and we contact in turn agents "on the left" and then the ones "on the right" to negotiate the best conditions (price, quality, etc.) needed to turn the specific link into a robust form.

Formal details of the agent protocol driving this approach are described in Section 4, whilst an example of its operation is found in Section 5.

4 Details of Protocol and Agent Decisions

In the previous section we have introduced a multi-phase negotiation mechanism to enable service composer agent to manage several negotiation processes (with providers of services s_i and s_j). The details described here relate to a single non-robust link only. Interdependencies between non-robust links are not considered in this work.

4.1 Phase 1

We start with the service composer agent calculating H_{T_i,T_j}.

Step 1: The first negotiation step comprises one-shot interaction between the service composer and all providers of services s_i, triggered by a Call-for-Proposals message from the service composer, which has H_{T_i,T_j} as its content. The negotiation protocol is shown in Figure 4 a). The type of response generated by service providers in the protocol is based on the following conditions.

a) `Proposal`: Each service provider will check if $\mathcal{T} \models Out\_s_i \sqsubseteq H_{T_i,T_j} \sqcap Out\_T_i$. If so, that provider will respond positively and the process will terminate.
b) `Refuse`: Alternatively, services for which $\mathcal{T} \models Out\_s_i \sqcap H_{T_i,T_j} \sqsubseteq \bot$ will be deemed unsuitable for further negotiation and their providers will refuse participating in the negotiation.
c) `Counter-Proposal`: If there is satisfiable intersection, i.e., $\mathcal{T} \not\models Out\_s_i \sqcap H_{T_i,T_j} \sqsubseteq \bot$, these providers will respond to the service composer with their output specifications Out_s_i.

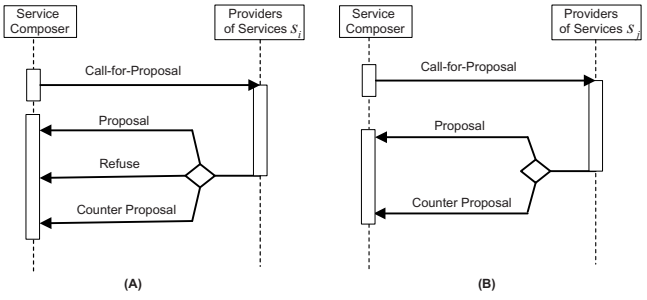

Fig. 4. Protocols to Support First Phase of Negotiation

Step 2: In case Step 1 ends up with counter proposals from providers of s_i, the service composer will use the negotiation protocol shown in Figure 4 b) to initiate negotiation with the providers of services s_j, sending them a Call-for-Proposals message with the set of all counter-proposals Out_s_i from Step 1 as its content. Step 2 can result in a robust composition if any of the services s_j has an input In_s_j which subsumes any of the counter-offers. The following response options are available to service provider agents.

a) `Proposal`: if $Sim_{\mathcal{T}}(Out\_s_i, In\_s_j)$ is of a PlugIn type, the agent responds positively;
b) `Counter-Proposal`: If $Sim_{\mathcal{T}}(Out\_s_i, In\_s_j)$ is an Intersection type, the agent responds with a counter-proposal which is the extra description required for this concrete semantic link $sl_{i,j}$ i.e., $\mathcal{T} \models Out\_s_i \sqcap H_{s_i,s_j} \sqsubseteq In\_s_j$. Below we refer to this as a "paired offer" between two provider agents.

If the second step ends up with counter-proposals rather than proposals, the composer will initiate the second phase of negotiation.

4.2 Phases 2 and 3

Step 1: In the second and third phase of negotiation the service composer uses the protocol shown in Figure 5, to solicit offers of using additional services (in Phase 2), or adapting or even developing services (in Phase 3), which can turn the particular link in its robust form. These phases build on the data about semantic fit gathered during Phase 1, in a way of a matrix linking service providers for s_j as rows and service providers for s_i as columns. The matrix, an example of which is shown in Table 1, contains the specific "paired offer" Extra Descriptions H_{s_i,s_j} in the respective cells, and the abstract Extra Description H_{T_i,T_j} elsewhere. The initial `Call-for-Proposals` message will refer to this matrix in its contents.

In response to the CFP the negotiation protocol presents the following response options.

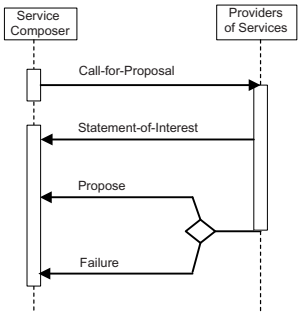

Fig. 5. Protocol to Support Second Phase of Negotiation

a) `Statement-of-Interest`: This is an optional response option. The service provider can send a `Statement-of-Interest` to buy time for finding other service providers (or coalition formation) that can help in delivering the required information.
b) `Proposal`: A service provider agent may use one of their services, or employ coalition formation techniques [21,22] and enlist a service from other agents for providing the needed additional specification. If these attempts succeed, the provider will responds positively with a proposal. For Phase 3, the service provider may propose adaptation of their service to provide the required specification, or the development of the extra filters required.
c) `Failure`: It is possible that the service provider agent is not able to form a coalition, in this case the service provider will responds with `Failure`.

Table 1. Matrix for Advertising Paired Services

	s_1^1	s_1^2	s_1^3
s_2^1	H_{T_1,T_2}	H_{T_1,T_2}	H_{T_1,T_2}
s_2^2	H_{T_1,T_2}	H_{s_3,s_2}	H_{T_1,T_2}

If any proposals are received, the service provider agent can terminate the protocol and select the best proposal. Otherwise they initiate Step 2.

Step 2: The service composer will initiate the negotiation with providers of services s_j using the same protocol (shown in Figure 5). The service providers are presented with the relevant specifications Out_s_i, gathered during Phase 1, and with the same response options. In Phase 2, the providers of services s_j should find extra services which can act in parallel with their offering to achieve a PlugIn relationship in relation to Out_s_i. In Phase 3, the providers should consider the costs and feasibility of adapting their services to handle the inputs specified by Out_s_i.

In case the protocol ends with a `Failure` then the negotiation proceeds to the third phase, where the service composer agent uses the same protocol template to check whether service providers are willing/able to develop new service that can complement their services for robust composition.

5 An Example Negotiation

Here we demonstrate how our approach can be applied to the example service composition covered in the Examples 1 to 3. We are focusing on the semantic link $sl^A_{1,2}$ in Example 1, which is non-Robust. The Service Composer Agent will calculate $H_{T_1,T_2} \equiv \forall netSpeed.Adsl1M$ (Example 2).

5.1 Phase 1

The Service Composer Agent will issue a CFP with an objective of H_{T_1,T_2} to all providers of candidate services for T_1, namely s^1_1, s^2_1, and s^3_1.

According to Example 3, we have $\mathcal{T} \models Out\_s^1_1 \sqsubseteq H_{T_1,T_2} \sqcap Out\_T_2$. Therefore the agent providing s^1_1 will respond with `Proposal` message, where they specify the conditions (price, QoS,etc.) for using their service. The agent providing s^2_1 will respond with `Refuse` message since, according to Example 3: $\mathcal{T} \models Out\_s^2_1 \sqcap H_{T_1,T_2} \sqsubseteq \bot$. And the agent providing s^3_1 will respond with `Counter-Proposal`$(Out\_s^3_1)$ since $\mathcal{T} \not\models Out\_s^3_1 \sqcap H_{T_1,T_2} \sqsubseteq \bot$ (Example 3).

Having received all three responses, the Service Composer Agent will choose s^1_1 and terminate the negotiation over this semantic link. If the system did not contain s^1_1 or its service provider agent did not send a response, the Service Composer Agent will take the payload of the Counter-Proposal message $Out\_s^3_1$ and send it as the objective of a CFP message to the agents providing services s^1_2 and s^2_2. Since $Sim_{\mathcal{T}}(Out\_s^3_1, In\_s^1_2)$ is of PlugIn matching type (Example 3), the provider of s^1_2 will respond with `Proposal` message, where they specify the conditions (price, QoS,etc.) for using their service.

This is not the case for the provider of s^2_2, where $Sim_{\mathcal{T}}(Out\_s^3_1, In\_s^2_2)$ is of Intersection matching type. This provider will calculate $H_{s^3_1,s^2_2}$ using the following: $\mathcal{T} \models Out\_s^3_1 \sqcap H_{s^3_1,s^2_2} \sqsubseteq In\_s^2_2$ which results in the $NetworkConnection$ to be $\forall netSpeed.Adsl1M \sqcap \forall netSpeed.AdslLimited$ (speed limited between $1mBytes$ and $1.5mBytes$).

The provider will then respond with this value as contents (payload) in a `Counter-Proposal` message.

Upon receiving all responses, the Service Composer agent will accept the best of all `Proposal` messages (the first and only one here). If such messages are not returned, the Service Composer agent will initiate the second phase of the negotiation, using the results $H_{s^1_1,s^1_2}$ from the first phase.

5.2 Phases 2 and 3

In the second phase, the Service Composer will ask the candidate service providers if they can provide an additional service (that provides the missing description H) to

ensure the semantic link is robust. The phase starts by the Service Composer issuing a CFP message to the providers of s_1, where the content of the message refers to the matrix shown on Table 1. The matrix will contain H_{T_1,T_2} for all pairs of candidate services apart from the cell (s_1^3, s_2^2), where the content will be H_{s_3,s_2} based on the "paired" counter-proposal reached at the end of Phase 1. Service providers will attempt to find additional services by potentially building coalitions and send either an agreement or a rejection. If the service composer does not receive any agreements, they will contact the two providers of s_2 with a CFP message, containing the output specifications received in the first phase of the negotiation. The two service providers will attempt to find an additional service to handle these output specifications, and respond accordingly. The third phase will repeat the interaction pattern of the second phase.

6 Related Work

We review some works related to our main contributions i.e., i) *Robustness* in semantic web service composition and ii) *Agent-based Negotiation* for service composition.

6.1 Robustness in Composition

An intuitive method [12] to immediately retrieve the *Extra Description* consists in discovering services that return this description as output parameters. Such a solution can be employed and implemented in any composition approach. In case of a non-robust semantic link, the *Extra Description* is exposed to a Web service discovery process which is in charge of retrieving relevant Web services. The latter services are able to provide the *Extra Description* as output parameters. The *Extra Description* can be reached by one or a conjunction of Web services, depending on the *Extra Description* and the discovery process. In contrast we use agent based negotiation for obtaining robust compositions of web services. This reduced the complexity of the whole approach by assuming agents interfacing sets of services that can resolve robustness of some semantic links. In more particular the proposed approach is of linear complexity i.e., each (distributed) agent only needs to look through several options/counter-offers.

Alternatively the set of Extra Descriptions is suggested to the end user in order to be relaxed in [23]. This user is then responsible of providing the Extra Description that the system needed to elaborate the final composition. The new information that end users will provide to the system is necessary to compute and elaborate a robust composition of web services, hence satisfying the initial user request. The suggested method has the advantage of relaxing constraints on the end user. In contrast we suggest an automated approach which does not require any end user support.

6.2 Agent-Based Negotiation

Agent-based approaches have recently been used to provide effective automated solutions to web service composition. This is partly because agent negotiations provide an effective way of addressing the complete issues associated with automated service composition [23]. Negotiation between software agents is one of the fundamental research issues in multi-agent systems. In this respect, this paper introduces several negotiation

processes that an be employed by agents to manage different issues within a service composition problem. The negotiation processes range from simple one-shot interactions to handling counter proposals across different processes and facilitating agent-based coalition-formation. The subject of coalition-formation is explored in [22] and agent-based coalition formation for service composition is discussed in [21]. In future we intend to focus on the coalition formation strategies and the trade-offs that can be offered to service provider agents within the service composition problem.

7 Conclusion

Ensuring robust semantic links between elements of composite services is very important in real scenarios of composition, and a mechanism to achieve this in an automated, effective and efficient fashion is needed for scalable and practical applications of web service composition. In this paper we propose such an automated approach which uses a formal model of semantic robustness, and agent-based negotiation protocol to ensure automation, effectiveness and efficiency. As shown by the example application and the specification of the approach, it can find automated solutions without involving humans, and also satisfy the criteria for effectiveness since innovative solutions can be found using coalition formation, and agents can customize services for lucrative opportunities of use. The dynamic nature of the negotiation protocol results in a number of requirements (in real-world scenarios) on the service providers side i.e., their willingness to create a new service on demand or to customize an existing service according to the composer's requirements. Finally, the approach is designed to ensure efficiency by exploring the free solutions first, then the low-cost use-based solutions, and only at last resort it considers service adaptation and development.

Our approach goes beyond the prevalent one-shot procedure (sending request and collecting results) by allowing agents to play a more active role in the composition process. The main direction for future work is to consider robustness in more expressive composition of web services (e.g., in case of conditional branching: multiple successors for on task with different input parameters). In addition, since running the negotiation protocol process during composition instantiation will affect the composition performance, some heuristics-based experiments on that specific point need to be driven. Finally optimization of robustness along web service composition needs to be investigated.

Acknowledgments

This work is conducted within the European Commission VII Framework IP Project Soa4All (Service Oriented Architectures for All) (*http://www.soa4all.eu/*), Contract No. IST-215219.

References

1. Bennett, K., Munro, M., Xu, J., Gold, N., Layzell, P., Mehandjiev, N., Budgen, D., Brereton, P.: Prototype implementations of an architectural model for service-based flexible software. In: Hawaii International Conference on System Sciences, vol. 3, p. 76b (2002)
2. Wielinga, B., Schreiber, G.: Configuration-design problem solving. IEEE Expert: Intelligent Systems and Their Applications 12(2), 49–56 (1997)

3. Motta, E.: Parametric Design Problem Solving - Reusable Components for Knowledge Modelling Case Studies. IOS Press, Amsterdam (1999)
4. Lécué, F., Mehandjiev, N.: Towards scalability of quality driven semantic web service composition. In: ICWS (2009)
5. Sycara, K.P., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. J. Web Sem. 1(1), 27–46 (2003)
6. Baader, F., Nutt, W.: The Description Logic Handbook: Theory, Implementation, and Applications (2003)
7. Ankolenkar, A., Paolucci, M., Srinivasan, N., Sycara, K.: The owl-s coalition, owl-s 1.1. Technical report (2004)
8. Fensel, D., Kifer, M., de Bruijn, J., Domingue, J.: Web service modeling ontology submission, w3c submission (2005)
9. Sivashanmugam, K., Verma, K., Sheth, A., Miller, J.: Adding semantics to web services standards. In: ICWS, pp. 395–401 (2003)
10. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3c recommendation, W3C (2004)
11. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 34–43 (2001)
12. Lécué, F., Delteil, A., Léger, A.: Applying abduction in semantic web service composition. In: ICWS, pp. 94–101 (2007)
13. McIlraith, S.A., Son, T.C.: Adapting golog for composition of semantic web services. In: KR, pp. 482–496 (2002)
14. ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of web services as parametric design. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 321–336. Springer, Heidelberg (2004)
15. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 385–398. Springer, Heidelberg (2006)
16. Horrocks, I.: Using an expressive description logic: Fact or fiction? In: KR, pp. 636–649 (1998)
17. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
18. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: WWW, pp. 331–339 (2003)
19. Lécué, F., Delteil, A.: Making the difference in semantic web service composition. In: AAAI, pp. 1383–1388 (2007)
20. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F., Mongiello, M.: Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an emarketplace. In: ECRA, vol. 4, pp. 41–50 (2005)
21. Muller, I., Kowalczyk, R., Braun, P.: Towards agent-based coalition formation for service composition. In: IAT 2006: Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology, Washington, DC, USA, pp. 73–80. IEEE Computer Society, Los Alamitos (2006)
22. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. 101(1-2), 165–200 (1998)
23. Hassine, A.B., Matsubara, S., Ishida, T.: A constraint-based approach to horizontal web service composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 130–143. Springer, Heidelberg (2006)