Service Composition in the Context of Grid

Xiaofeng Du, William Song, Malcolm Munro
Computer Science Department, University of Durham
DH1 3LE, Durham, UK
{xiaofeng.du, w.w.song, malcolm.munro}@dur.ac.uk

Abstract

Grid computing has become an important new research field. The goal of the Grid computing infrastructure is to pervasively access the computational resources available on the Internet. The web service technology has the potential to achieve the goal of grid computing because of its self-contained, self-describing, and loosely coupled features. The resources can be discovered and shared through the web service’s process interface. In this paper we discuss how the web services can be composed from a conceptual perspective and propos a context based semantic model for better describe web service in order to realize automatic service discovery, composition, and invocation.

1. Introduction

Current researches on the World Wide Web and the Grid, as well as Web Services, have come to a consensus issue. That is, how to extract and represent semantics for the Web information and the Grid recourses. As known to us, the World Wide Web provides an infrastructure for information exchange and interoperation and the Grid provides an infrastructure for resource sharing and cooperation [2]. The interoperability and collaboration requires common understanding of information/data and resources and the understanding requires canonical and well-formed semantic description. The semantic description is used so that the human and machine can understand each other [15].

The context of application of the Web and the Grid is e-Science, which studies how computer and communication technology can support and enhance the scientific process by enabling scientists to generate, analyse, share and discuss their insights, experiments and results in a more effective manner [16]. The Grid is the underlying computer infrastructure that provides these facilities. Web Service, realized in Grid services, uses a standardized XML messaging system and a stack of communication protocols to makes software available over the Web [1]. One aspect of its significance lies in composition of software components available on the Grid to form an application/service for users to accomplish certain tasks. How to discover, analyse, form, and compose resources and services for users’ need depends strongly on the understanding of meanings of resources/services. In other words, semantic description for the resources/services is indispensable in development of useful and powerful Web/Grid based applications. A well defined semantic description model can enable automatic service discovery, invocation, and composition.

2. Web Service, Grid Service and Grid Infrastructure

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network that has an interface described in a machine-processable format (specifically WSDL [4]) [17]. Grid services are defined by the Open Grid Services Architecture (OGSA) [3] as mechanisms for creating, managing, and exchanging information among entities. A Grid service is a Web service that conforms to a set of conventions (interfaces and behaviours) that define how a client interacts with a Grid service.

In spite of the above definitions, we can put services into the following types [5]:

1. To exchange information and documents;
2. To interactively and cooperatively perform functions;

3. To share available resources.

The first type is the Web systems or software for information exchange. For the second one, we consider that web service is a chain of (sub-) services in order to accomplish a user given task. The services in the chain are loosely coupled. Based on requirements, best match or suitable match is found between two services. If our task is to calculate \((a*x+b*y)\), then we assume that the functions * (multiply) and + (addition) are available on the Web. A web service to accomplish this task is to find the related sub-services, apply the parameters, compose the services, and return the result, using various protocols, and specifications languages [6]. Figure 1 illustrates the relationship between task and services, and how the services can be formed into a service flow to achieve the task.

Grid service is considered to be of the third type, which emphasizes cooperation and sharing of resources (including data) among entities [2]. In particular, to share and interoperate various resources, e.g. CPU time, storage, files, from physically different nodes (entities) in a grid to form a virtual organization. For example, a virtual music album can be created through linking together music pieces from different websites or PCs.

In the grid infrastructure, the resource sharing is task driven [2]. A resource consumer issues tasks to request resource sharing from the resource providers. The web services can be considered as an interface between tasks and resources, which can consume suitable resources to achieve a certain task. In figure 2, the diagram illustrates the relationships among resource consumer, task, web service, resource, and the grid infrastructure. Under the grid infrastructure, resource consumer issues tasks; the tasks motivate the design of the web services; the web services consume resources to achieve tasks.


Figure 2: Relationships among task, web service, resource, and grid infrastructure

With an exponentially increasing amount of information, documents, resources, and services available on the Web, to find a good and reasonable match between the users’ requirements and the capabilities of Web/Grid services as well as to form a suitable composition out of the service components to serve a demanded activity is a prominent problem. That is because we lack an effective and efficient means to describe services, components, and objects existing in the Web.

3. Conceptual Service Composition

The web services published on the Internet are mostly atomic services which only provide simple and primitive functions. Therefore, if a service provider wants to provide better services or promote more efficient sharing of resources, publishing composite services is a sound solution. Service composition can be done through identifying sub tasks, locating suitable sub-services, and formatting the sub-services into a service flow, and executing the service flow to achieve a task which is the goal of the composite service.
The difficult steps to be realized in the process of service composition are locating suitable sub-services and formatting sub-services into a service flow. Locating suitable services will be discussed in next section to see how the semantic web technology can benefit service selection. This section will focus on what aspects need to be considered when formatting the selected sub-services into a service flow by linking relevant sub-service together.

When you link two services together, in fact you are linking the former service’s outputs with later service’s inputs, i.e. the later service’s inputs can be inputted by the value from the former service’s outputs. This linking is based on the compatibility of the input and output data type, satisfiability of per and post conditions, and the similarity of the inputs and outputs’ semantics. These are the basic criteria that need to be considered before joining two services together. However, for sufficiently achieving a real task, just consider the basic criteria are not enough, some other contextual information also need to be considered [7], such as:

- **Time**: When two services are running in parallel, they have to make sure they return the outputs at the same time before their outputs go into next service’s inputs.

- **Data Consistency**: One service’s output and another service’s input have the same data type and semantic meaning does not mean the data consistency is satisfied. For example, one service’s output is a weight in gram, but another one’s input needs a weight in kilogram. In this example both service’s input and output data type are ‘double’ and both of their semantic meaning are ‘weight’, but the input and output of these two services cannot be directly linked together.

- **Location**: Sometimes the location of a service also can affect the service composition. For example, one service requires an address as an input and another service can return an address, but there is a case that these two services cannot be linked together. That is when one service is in UK and another is in US.

Once a composite service has been built, the service provider needs to publish its inputs and outputs as public interfaces for the service requester to invoke. The inputs and outputs of a composite service are generated from some of its internal sub-services’ inputs and outputs. If we use a directed graph $G(V, E)$ to represent all the services in the grid environment and their possible relationships\(^1\), then a composite service can be represented as a sub-graph of $G$.

$$Sub_G(V_1, E_1) \subseteq G$$

The $Sub_G_1$ together with its context can be represented as another sub-graph of $G$.

$$Sub_G_2(V_2, E_2) \subseteq G$$

$$Sub_G_1 \subseteq Sub_G_2$$

Then we can represent the context of the composite service in the grid environment, which is difference between $Sub_G_2$ and $Sub_G_1$:

$$C(V_1, E_1) = Sub_G_2 - Sub_G_1$$

The set of arcs $E_c$ in $C$ represents the inputs and outputs of the composite service.

$$E_c = I \cup O$$

Where

- $I$: A set of inputs of the composite service.
- $O$: A set of outputs of the composite service.

![Figure 3: Composite Service, Context and Grid](image)

Figure 3 shows a graphical illustration of the conceptual representation of the relationship among composite service, its context, and the grid environment. In the diagram, the red arrows represent the internal relationships between sub-

\(^1\) Note: the arcs in the graph $G$ are uncertain due to different composition requirements. Here just captures a possible situation.
services inside the composite service, the black arrows represent the inputs and outputs of the composite service, and the blue dashed arrows represent the relationships between the services in the grid environment.

4. Semantic Service Description Aspects and Matchmaking

As mentioned in the previous section, one of the difficult steps to be realized in the process of service composition is locating suitable sub-services. The reason is because the current web service description technology, e.g. WSDL, only provides syntactic description of a service rather than semantic description [8]. A WSDL description does not provide the information about what the service can do, thus a user has to read an extra description to get the functionalities of the service. On the other hand, the web service global registry UDDI only provide keyword matching searching rather than semantic searching, thus, it is difficult for a user to accurately find a suitable service. Therefore, integrating semantics into the current web service description and discovery technologies is crucial to improve the usability of web services and to achieve automatic service composition and resource sharing under grid infrastructure.

4.1 Semantic Service Description Aspects

A most important issue in semantic description for objects is the correct and exact capture of semantics. Lara et al. [10] and Fensel et al. [9] have discussed the semantic description requirements for describing a web service’s capability, such as pre-condition, post-condition, textual description, services, and identifier. However, only integrating semantics is not sufficient to fully address a service, the context relevant information about a service cannot be ignored when describing a service. A semantic definition in a dictionary manner is not feasible because vague meaning is not only indispensable but necessary for better understanding as well. It is also difficult to provide all possible circumstances or instances of a concept (of an object), if not impossible. Actually, in most cases, people understand each other in a certain context. For example, when we talk about “jaguar” together with “BMW” and “Volvo”, it is almost for certain that we mean a car instead of an animal. Using a contextual model, we can better identify the meaning of a given object and therefore select a most suitable service or information object.

Therefore, we have developed our initial contextual based semantic service description model which integrates the context together with semantics to better describe a service [7]. This model addresses a service from six aspects:

1. IOPE: This aspect addresses the service’s input data type, output data type, pre-condition, and effects. It also addresses the semantic meaning of the inputs and outputs.

2. Metadata: This aspect addresses the non-functional description of the service including identifier, natural language description, location, quality attributes, and category.

3. Ontology: This aspect addresses the concept and domain of the service.

4. Upper Compositionality: This aspect addresses which kind of services can be composed by using this service. This aspect indicates the relationships between this service and other services.

5. Lower Compositionality: This aspect addresses what kind of services can be used to compose this service. This aspect also indicates the relationships between this service and other services.

6. Resource: This aspect addresses what kind of resources the service will consume. The resources include renewable resources, such as bandwidth, memory allocation, and CPU usage, and consumable resources [11], such as time. The reason for considering the resources in a service description is because the processes require resource and the processes decide how the service behaves. Therefore, the resources information can be an important criterion to identify a service. However, if a service provider thinks the resources allocation information could be private, he can hide this information from the service requesters.

Figure 4 illustrates the relationship between the six aspects for addressing a service.
The notation used in figure 4 is:
1) \( S_i \rightarrow \text{Service}_i \)
2) \( \text{Service}^i \) can be either the parent or ancestor of \( \text{Service}_i \)
3) \( S_1, S_2, S_3, \) and \( S_4 \) are any services.
4) \( I, P \) is the Inputs and Pre-condition, and \( O, E \) is the Outputs and Effects.

### 4.2 Matchmaking

Based on the contextual based semantic model, we propose a semantic similarity computation approach to compare and measure the semantic similarity between a requested service and a candidate service [6]. This approach contains three steps. The first step is to compare individually the requested service (which is expanded to a sequence of services) and the candidate services from the service pool. The result is a set of ranked candidate service nodes. In the second step, we search for those candidate service pairs that meet the service pairs in the requested service sequence. The result is a set of ranked candidate service pairs. In the last step, the service pairs in the candidate service pair set are assembled into a number of sequences of services according to the requirements of the requested service sequence. In the following, we discuss in detail these three comparison and matchmaking steps.

**Node comparison.** To find a semantic match of two services \( n \) and \( m \) we compare their contextual characteristic sets \( \lambda(n) \) and \( \lambda(m) \) based on the contextual based semantic model. We have developed three semantic matchmaking methods for \( \lambda \). In order to reflect the semantic complexity of contextual based semantic descriptions for services and domain knowledge, we also assign a set of weights to the semantic characteristic set \( \lambda() \). For a given service \( s \) in the service flow, we get a set of pairs \((s_k, n_k)\), where \( s_k \) is a service from the service pool, and \( n_k \) is an associated number to indicate how close the service \( s_k \) is semantically to \( s \). When there are a large number of candidate services, we need to set a threshold value to contract the candidate set to a reasonable size.

**Pair comparison.** The second step is semantic comparison and matchmaking for service pairs. We consider two adjacent nodes, \( t_i-t_{i+1} \), in the requested service sequence. After the first step we got two candidate service sets, \( S_i \) for \( t_i \) and \( S_{i+1} \) for \( t_{i+1} \). Using the service composite definition, we can generate a number of service composites \( s_i-s_j \), where \( s_i \in S_i \) and \( s_j \in S_{i+1} \). The services in each pair are restrained by the interdependent relationship in terms of their IOPE. Simply speaking, if there is a suitable IOPE interdependence (or match) between two service \( s_i \) and \( s_j \) and a semantic inclusion relation between \( t_i \) and \( s_i \) and \( t_{i+1} \) and \( s_j \) respectively, the pair \( s_i-s_j \) is a semantically compatible candidate pair matching the pair \( t_i-t_{i+1} \). Similarly, the ranking and threshold mechanisms are applied to reduce the size of the candidate pair set.

**Sequence comparison.** The last step is sequence semantic matchmaking, where we create a sequence of services from the above-obtained candidate service pairs. Through iteratively applying the definitions given above, we get a number of candidate service sequences that match the requested sequence. We select the one best fit the need. Here we should emphasize the important role that the service scheduling plays in this step. When all candidate services are dynamically coupled and composed, it is possible that some sequences may contain services which suddenly become unavailable and require instantaneous replacements. In addition an important step in the service scheduling process is to quantitatively measure semantic distance between candidate services, service pairs, service sequences and given service sequence (task) and its components (such as given tasks and task pairs).

### 4.3 Semantic distance computation

Based on the semantic characteristic functions, we develop a set of semantic measurement methods to measure semantic distances between two service nodes. We define that a semantic distance \( dist \) between two nodes \( S \) and \( R \) as
Here $\lambda$ is the set of all the semantic characteristics functions.

The semantic measurement methods include a number of weighted formulas for computing the semantic distances between two service nodes, between two service pairs, and between two service sequences:

1. $\text{semantic\_distance}(S, R) = \omega_s \cdot \text{dist}(\lambda(S), \lambda(R))$
2. $\text{semantic\_distance}(S_1-S_2, R_1-R_2) = \omega_p \cdot \text{dist}(\lambda(S_1-S_2), \lambda(R_1-R_2))$
3. $\text{semantic\_distance}(S_1-\ldots-S_n, R_1-\ldots-R_n) = \omega_f \cdot \text{dist}(\lambda(S_1-\ldots-S_n), \lambda(R_1-\ldots-R_n))$

Here $\omega_s$, $\omega_p$, $\omega_f$ are the weights for the individual, pair, and sequence semantic computations respectively.

A simple example is given below to illustrate how the algorithm works.

In order to construct a composite service to calculate the area of trapezium, one of the required services, an addition service, has to be located. The specification for this service is listed below:

- Number of inputs: 2,
- Input data type: double,
- Output data type: double,
- Description keyword: addition,
- A mathematics calculation service.
- Can be used to calculate the perimeter of a rectangle

By applying the algorithm, we get the results which have been listed in table 1:

<table>
<thead>
<tr>
<th>Matching Aspects</th>
<th>Service1 double addition(double a, double b)</th>
<th>Service2 int addition(int a, int b)</th>
<th>Service3 double power(double a, double b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOPE</td>
<td>1.0</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Ontology</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Metadata</td>
<td>0.667</td>
<td>0.667</td>
<td>0.0</td>
</tr>
<tr>
<td>Upper Compositionality</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Sum</td>
<td>3.667</td>
<td>3.167</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Table 1: Semantic Matching Result of an Addition Service

The values in the table indicate the candidate services satisfaction rate for each aspect in the service requirement specification. From the above result, it is easy to see that the Service1 has the highest overall satisfaction score which represents the shortest semantic distance to the requirement, so the Service1 will be the chosen service.

Related Work

Fensel et al. [9] proposed a web service modelling work (WSMF) to address how a service should be described. Roman et al. [12] proposed a web service modelling ontology (WSMO) based on the WSMF to semantically describe the aspects proposed in WSMF for describing services. Martin et al. [13] proposed a semantic mark-up language OWL-S to describe web services. This language describes a web service through three components: service profile, service process model, and service grounding.

Medjahed et al. [14] provided a whole solution from semantic service description, service composability model, to automatic composition of web services. However, as we discussed previously, these research efforts did not consider enough contextual aspects for describing a service which is important in the process of service discovery and composition.

Conclusion and Future Work

Defining a contextual based semantic model to analyze and describe the structure and characteristics of services is very important in the research areas of semantic web service, grid computing, and web services. In this paper, we discussed how the web services can be composed from a conceptual perspective, proposed a contextual based semantic description model for automatic web service discovery and composition, and introduced a semantic matchmaking algorithm based the model to
accurately identify a service or a sequence of services. This is particularly significant in semantic based automatic service searches and matches as the number of services on the web is growing exponentially. Our next step is to develop an advanced quantitative analysis algorithm for semantic matching and to evaluate its performance. In order to achieve automatic service discover and composition, we also need to consummate the contextual based semantic model to more sufficiently describe services.

References

17. W3C Web Service Definition http://www.w3.org/TR/ws-arch/