

Characterizing Services Composeability and OWL-S based Services Composition

Zhonghua Yang¹, Zhang Jing Bing², Jiao Tao¹ and Robert Gay¹

¹Information Communication Institute of Singapore
School of Electrical and Electronics Engineering
Nanyang Technological University, Singapore 639798

²Singapore Institute of Manufacturing Technology
71 Nanyang Drive, Singapore 638075
eZhYang@ntu.edu.sg

Abstract. Grid has emerged as a new paradigm for integration within dynamic virtual enterprises. Given a service-oriented Grid environment, more complex, value-added sophisticated services and applications can be built via service composition. In this paper, we discuss Grid services composition by leveraging Semantic Web services standards and technology, especially OWL-S. The characterization of service composeability is presented in the context of OWL-S. The techniques for constructing the Web/Grid service composer are described, and the focus is placed on composition engine and OWL-S Matcher.

1 Introduction

Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic “virtual organizations (VOs)”. In the recent years, there is the alignment of Grid technology to Web services, and a *Grid services* is defined as a *Web service* that is designed to operate in a Grid environment and meets the requirements of the Grid(s) in which it participates. The early definition that a Grid service is a service that implements the GridService portType of OGSF is now considered deprecated. Web services are assumed widely shared and available in the community. One of the primary benefits of service-oriented architecture (SOA) is the ability to compose applications, processes, or more complex services from other less complex services. This activity is called *service composition*. Service composition is emerging as the technology of choice for building cross-enterprise applications on the Web. Given the well-defined (and well-described) interfaces, loose-coupling, and Web standard-based interaction protocols between services, a complex value-added service or process can be dynamically constructed out of available Web services as needed.

In this paper, we present a Semantic Web services based approach to Grid service composition. Service semantics is represented in OWL-S. Based on OWL-S, we characterize the composeability of Web/Grid services on which OWL-S based

grid services composition is presented, and a OWL-S based Web/grid services composer is described, focusing on composition engine and OWL-S matcher.

2 Composeability of Web services

Composeability of Web services refers to the ability that participant services can actually work together as expected. In a sense, composeability of Web services is the extent to which two services are related in a way that they can be assembled to offer a value-added service.

In view of service composition, it is necessary to decide when one Web service *matches* what we want. It is well known that these characterization of a function or a software component cannot be determined by its signature alone. Web services composition needs to characterize the dynamic behavior, i.e., semantics, of each Web service.

There are different kinds of semantic matches. While *exact match* between services (or between service and requirements) appears the best choice, many flavors of relaxed match may also serve the purpose. In the remainder of this section, we attempt to characterize matches based on semantics. This characterization requires the semantic description of Web services in OWL-S [1]. OWL-S is a OWL-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. There are two aspects of behavior that can be described in OWL-S: the information transformation represented by *inputs* and *outputs* and the state change produced by the execution of the service, represented by *preconditions* and *effects*. This together is known as IOPE.

Inputs and outputs are named and typed using either OWL-S ontologies (classes or properties) or data types that XML Schema provides. OWL-S allows the specification of the set of conditions under which the outputs or the effects may result. In addition, OWL-S process ontology allows to associate conditions with outputs and effects because a service's outputs and effects are often predicated on some observable characteristic of the system.

The following characterization is based on OWL-S IOPE description of services.

Definition 1 (Inputs-Match).

$$match_{inputs}(A, B) = A_{inputs} \Leftrightarrow B_{inputs}$$

Service A is said inputs-matched with service B if and only if each input of service B has corresponding matched partner in service A's inputs.

If Service C is looking for a service B for composition, any advertised service A that is input-matched with B can be used. Because we do not have the precondition term, there is no guarantee that the preconditions of the advertised service actually holds, so we may have to provide an additional "wrapper" to

establish the required precondition before we invoke the advertised service. The Guarded Inputs-Match takes into account the required preconditions.

Definition 2 (Guarded Inputs-Match).

$$match_{guarded-inputs}(A, B) = A_{inputs} \Leftrightarrow B_{inputs} \wedge A_{pre} \Rightarrow B_{pre}$$

Service A is said Guarded inputs-matched with service B if and only if each input of service B has corresponding matched partner in service A's inputs under the same or weak precondition.

Definition 3 (Outputs-Match).

$$match_{outputs}(A, B) = A_{outputs} \Leftrightarrow B_{outputs}$$

Service A is said outputs-matched with service B if and only if each output of service B has corresponding matched partner in service A's outputs.

If Service C is looking for a service B for composition, any advertised service A that is output-matched with B can be used.

Definition 4 (Guarded outputs-Match).

$$match_{guarded-outputs}(A, B) = A_{outputs} \Leftrightarrow B_{outputs} \wedge B_{effects} \Rightarrow A_{effects}$$

Service A is said Guarded outputs-matched with service B if and only if they are output matched under the same or stronger effects.

The characterization of input-match and output-match is useful and brings flexibility for service composition. For example, sometimes we are unable to find an exactly matched service, we can find a service which is the combination of input-matched service and an output-matched service.

Definition 5 (Exact-IO-Match).

$$match_{exactIO}(A, B) = match_{inputs}(A, B) \wedge match_{outputs}(A, B)$$

If two services A and B produce the same outputs when supplying the same inputs, then two service A and B have the equivalent behavior in terms of their inputs and outputs.

A more complex service can be composed of from service C, D, and B, if we are looking for a service that has inputs of service D and outputs of C. In this case, the service B is “exact IO matched” with what we are looking for.

Definition 6 (Exact IOPE Match).

$$match_{iope}(A, B) = match_{exactIO}(A, B) \wedge A_{pre} \Leftrightarrow B_{pre} \wedge A_{effect} \Leftrightarrow B_{effect}$$

Service A is said IOPE-matched with service B if and only if each aspect of IOPE of service B's description has corresponding matched partner in service A's IOPE.

IOPE matched two services (one advertised and one requested) are essentially equivalent and thus completely interchangeable. Anywhere that one service is used, it could be replaced by the other with no change in observable behavior.

Exact IOPE match or Equivalence is a strong requirement. Sometimes a weaker match is “good enough.”

Definition 7 (PlugIn match).

$$match_{plugIn}(A, B) = A_{output} \subset B_{output}$$

An advertised service A subsumes a requested service B if and only if outputs of B are the subset of outputs of service A.

This match acknowledges that there is a weaker relation between two services. In other words, the advertised service A could be plugged in place of the requested service B [4, 2]. For example, a service that provides (any type of ...) “Books” could be of use for another service that expects “Computer Books”.

In some cases, the plugIn relation only holds for values of the input allowed by the precondition. The guarded plug-in match adds A_{pre} as an assumption (or “guard”) to exclude such cases.

Definition 8 (Guarded PlugIn match).

$$match_{GuardedplugIn}(A, B) = match_{plugIn}(A, B) \wedge A_{pre}$$

An advertised service A plug in matched with a requested service B under precondition of service A.

Definition 9 (subsume match).

$$match_{subsume}(A, B) = B_{output} \supset A_{output}$$

A service B subsumes match an advertised service A if and only if outputs of B are the superset of outputs of service A.

If a service B subsumes a service A, then the provider of service A does not completely fulfill the request for service B. The requester may use the provider to achieve its goals, but it likely needs to modify its plan or perform other requests to complete its task.

Similar to guarded plugIn match, we have the guarded subsume match.

Definition 10 (Guarded subsume match).

$$match_{Guardedsubsume}(A, B) = match_{subsume}(A, B) \wedge A_{pre}$$

A requested service B subsumes an advertised service A under precondition of service A.

In some cases, we are more concerned with the relation between the preconditions and effects of services. This is Pre/Effect matches which relate the preconditions of each service and the effects of each service.

Definition 11 (PE match).

$$match_{PE}(A, B) = B_{pre} \Leftrightarrow A_{pre} \wedge A_{effect} \Leftrightarrow B_{effect}$$

A service B is PE matched with any advertised service A if and only if both services produce the equivalent effects under equivalent preconditions.

Definition 12 (PlugIn PE match).

$$match_{PlugIn-PE}(A, B) = B_{pre} \Rightarrow A_{pre} \wedge A_{effect} \Rightarrow B_{effect}$$

A service B is PlugIn PE matched with any advertised service A whose precondition is weaker (to allow at least all of the conditions that B allows) and whose effects is stronger (to provide a guarantee at least as strong as B).

Often, we are concerned with only the effects of services; thus, a useful relaxation is to consider only the effect part of the service description. Most preconditions could be satisfied by adding an additional check before consuming the service.

Definition 13 (Effect match).

$$match_{effect}(A, B) = A_{effect} \Leftrightarrow B_{effect}$$

A service B is effect matched by any advertised service A if and only if both services present the equivalent effects.

Definition 14 (PlugIn Effect match).

$$match_{plugIneffect}(A, B) = B_{pre} \Rightarrow A_{pre} \wedge match_{effect}(A, B)$$

A service B is plugIn effect matched by any advertised service A whose precondition is weak and both services present the equivalent effects.

We have characterized the composeability of a service by defining a variety of matches. Which match is most appropriate for composition will depend on the particular situation. Clearly, the choice of match depends on the context in which the match is used: how strong of a guarantee is needed about the relation between the two services in terms of their semantic description (for OWL-S, IOPE)? While an exact match appears attractive, a various weak form of match may be found serving the purpose.

3 Semantic-enhanced Grid Services Composition

Based on our characterization of services composeability, we present a composer of Web/Grid services which are described using Web services upper ontology OWL-S. The architecture of the composer (SWSC) is shown in Figure 1. The composer consists of the following main components: *OWL-S services repository*,

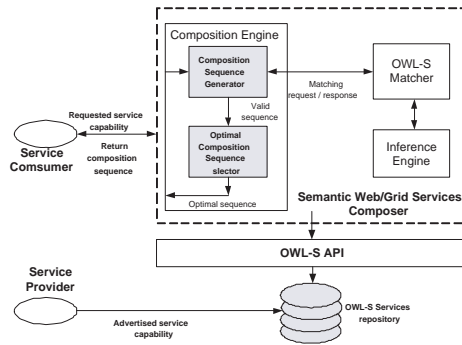


Fig. 1. Architecture for Grid services composition based on OWL-S

Composition engine: The composition engine actually accomplishes the composition based on the requirements from service consumer in terms of service capability represented as OWL-S IOPE. *OWL-S matcher* OWL-S Matcher plays the role of matchmaking between requested service capability and advertised service capability in the repository, and the matcher relies on the OWL Inference engine to perform semantic-based matchmaking. *OWL-S API library* [3]. It is a convenient programming library for implementing OWL-S based composer.

4 Conclusion

In this paper, we have discussed Grid services composition by leveraging Semantic Web services standards and technology, especially OWL-S. The characterization of service composeability is presented in the context of OWL-S. It is expected that this characterization facilitates more effective Grid service composition. A business process driven approach is briefly discussed to derive IOPE-based requirements of service provision. The techniques for constructing the Web/Grid service composer are described. Based on IOPE-based requirements, service composition is gradually generated with a forward or backward chaining of services.

References

1. David Martin, Mark Burstein, and Jerry Hobbs *et al.* *OWL-S: Semantic Markup for Web Services*. W3C Member Submission, 22 November 2004.
2. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic Matching of Web Services Capabilities. In *Proceedings of the First International Semantic Web Conference, LNCS 2342*, pages 333–347. Springer-Verlag, 2002.
3. Evren Sirin. OWL-S API. The MINDSWAP Group, University of Maryland, <http://www.mindswap.org/2004/owl-s/api/index.shtml>, 2005.
4. Amy Moormann Zaremski and Jeannette M. Wing. Specification Matching of Software Components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(4):333–369, October 1997.