Semantic Service Mediation

Liangzhao Zeng[1], Boualem Benatallah[2], Guo Tong Xie[3], and Hui Lei[1]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
{lzeng, hlei}@us.ibm.com
[2] School of Computer Science and Engineering University of New South Wales, Sydney, Australia
boualem@cse.unsw.edu.au
[3] IBM China Research Laboratory
Building 19, Zhongguancun Software Park, ShangDi, Beijing, 100094, P.R. China
xieguot@cn.ibm.com

Abstract. The service mediation that decouples service interactions is a key component in supporting the implementation of SOA solutions cross enterprises. The decoupling is achieved by having the consumers and providers to interact via an intermediary. The earliest service mediations are keyword and value-based, which require both service providers and consumers to adhere same data formats in defining service interfaces and requests. This requirement makes it inadequate for supporting interactions among services in heterogeneous and dynamic environments. In order to overcome this limitation, semantics are introduced into service mediations, for more flexible service matchings. In this paper, we proposed a novel semantic service mediation. Different from existing semantic service mediations, our system uses ontologies not only for one-to-one service matchings, but also for one-to-multiple service matchings. By performing service correlation systematically as part of the service mediation, services can be composted automatically, without any programming efforts (neither composition rules nor process models). We argue that a service mediator like ours enables more flexible and on-demand mediation among services.

1 Introduction

The service mediation enables decoupling among the service providers and consumers, which is a key component in Enterprise Service Bus (ESB) for supporting SOA solutions cross enterprises. Typically, the service mediation system contains three roles: (1) *service providers*, who publish services; (2) *service consumers*, who request services, (3) *service mediators*, who are responsible for service repository management, service matching, service invocation and invocation result delivery. The earliest service mediations are keyword and value-based (e.g., UDDI [6]). There are two major limitations in such service mediations: (i) the service discovery is keyword-based; (ii) service invocations are based on value of exchanged messages. For example, a service request is about retrieving a sports car's insurance quote, where the input parameter is `SportsCar` and output parameter is `CarPremium`. For the value-based service mediation, only the services that have input parameter `SportsCar` and output parameter `CarPremium` can match the request. In case service request and service interfaces' input/output parameters are not exactly matched, then data format mapping needs to be provided.

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 490–495, 2006.

Consequently, as an improvement to keyword and type-based solutions, semantics are introduced into service mediations [1,5], wherein ontologies enable richer semantics in service publication and more flexible matchings. However, current semantic service mediations only support one-to-one service matchings. Any sophisticate one-to-multiple services matchings (i.e., composing a collection of services to fulfill a service request) requires either defining knowledge base (e.g., composition rules) or creating process models [7,8]. In order to overcome these limitations, in our semantic service mediation, one-to-multiple matchings are enabled using correlation-based composition, by utilizing the semantics derived from service interface definitions. In particular, proposed correlation-based composition is transparent to service consumers, i.e., requires neither defining service composition knowledge bases nor creating process models. It should be noted that our correlation-based solution is complementary to those knowledge-based and process-based solutions. With our correlation-based solution, either knowledge-based or process-based service composition developers can focus on high level business logic to develop composition services, without understanding extraordinary details of service interfaces. In the case of the expected service interfaces that defined in knowledge bases or process models are not currently available, the service mediator can locate multiple services and correlate them to a "virtual service" to fulfill the service request.

The remainder of this paper is organized as follows: Section 2 introduces some important concepts. Section 3 presents the overview of the semantic service mediation. Section 4 discusses the correlation-based service composition. Finally, Section 5 discusses some related work and Section 6 provides concluding remarks.

2 Preliminaries

In our system, we adopt an object-oriented approach to the definition of ontology, in which the type is defined in terms of *classes* (See Definition 1) and an instance of a class is considered as an *object* (See Definition2). It should be noted that this ontology formulation can be easily implemented using OWL [4] and IODT [3]. We will present details on how to use ontologies to perform semantic matchings and correlation matchings in following sections.

Definition 1 (*Class*). A class C is defined as the tuple $C = \langle N, S, P, R \rangle$, where

- N is the name of the class;
- S is a set of synonyms for the name of class, $S = \{s_1, s_2, ..., s_n\}$;
- P is a set of properties, $P = \{p_1, p_2, ..., p_n\}$. For $p_i \in P$, p_i is a 2-tuple in form of $\langle T, N_p \rangle$, where T is a basic type such as integer, or a class in an ontology, N_p is the property name. p_1 ($p_1 \in P$) is the key property for identification;
- R is a set of parent classes, $R = \{C_1, C_2, ..., C_k\}$. □

In the definition of class, the *name*, *synonyms*, and *properties* present the connotation of a class; while *parent classes* specify relationships among the classes, i.e., present the denotation of a class. A class may have parent classes for which it inherits attributes. For example, class `sportsCar`'s parent class is `Car`, so the class `sportsCar` inherits all the attributes in class `Car`.

Definition 2 (***Object***)**.** An object o is a 2-tuple$\langle N_c, V\rangle$, o is an instance of a class C, where

- N_c is the class name of C;
- $V = \{v_1, v_2, ..., v_n\}$, are values according to the attributes of the class C. For $v_i \in V$, v_i is a 2-tuple in form of $\langle N_p, V_p\rangle$, where N_p is the property name, V_p is the property value. □

A service interface is denoted as $I_s(P_{in}, P_{out})$, where P_{in} ($P_{in} = \langle C_1, C_2, ..., C_n\rangle$) indicates input parameter classes, and P_{out} ($P_{out} = \langle C_1, C_2, ..., C_m\rangle$) indicates output parameter classes. An example of a service s's interface can be I_s ($P_{in}\langle$`SportsCar`$\rangle$, $P_{out}\langle$`CarInsurance`, `CarFinance`$\rangle$), which contains one input parameter and two output parameters.

A service request is denoted as $Q(O_{in}, E_{out})$, where O_{in} ($O_{in} = \langle o_1, o_2, ..., o_n\rangle$) indicates input objects, and E_{out} ($E_{out} = \langle C_1, C_2, ..., C_m\rangle$) indicates expected output parameters from the services. An example of a service request can be $Q(O_{in}\langle$`car`$\rangle$, $E_{out}\langle$`CarInsurance`, `CarFinance`$\rangle$), which contains one input object `car` and expects a service provides two outputs: `CarInsurance` and `CarFinance`.

Table 1. Examples

Entity	Example
service request	$Q_1(O_{in}\langle$`sportsCarA`$\rangle$, $E_{out}\langle$`CarInsurance`, `CarFinance`$\rangle$)
candidate service's interface	I_s ($P_{in}\langle$`Car`$\rangle$, $P_{out}\langle$`CarInsurance`, `CarFinance`$\rangle$)
interface set	$\mathbb{I}_k$={I_1, I_2}, where I_1 ($P_{in}\langle$`Car`$\rangle$, $P_{out}\langle$`CarInsurance`$\rangle$), I_2 ($P_{in}\langle$`Car`$\rangle$, $P_{out}\langle$`CarFinance`$\rangle$)

3 Service Matching in Semantic Service Mediation

By introducing ontologies into the service mediation, other than *exact matching*, we extend the service matching algorithm with two extra steps: *semantic matching* and *correlation matching*. Therefore, three steps are involved in our matching algorithm:-

Step 1. Exact Matching. The first step is to find exact matches, which returns service interfaces that have exactly the same parameter (input and output) classes as the service request;

Step 2. Semantic Matching. The system searches service interfaces that have parameter classes that are semantically compatible with the service request. In our system, the semantic matching is based on the notion of *Semantic Compatibility*.

Definition 3 (***Semantic Compatibility***)**.** Class C_i is semantically compatible with class C_j, denoted as $C_i \stackrel{s}{=} C_j$, if in the ontology, either (i) C_i is the same as C_j (same name or synonym in an ontology) , or (ii) C_j is a superclass of C_i. □

By adopting the definition of semantic compatibility, we say a class C semantically belongs to a class set $\mathbb{C}$ (denoted as $C \in_s \mathbb{C}$) if $\exists C_i \in \mathbb{C}$, $C \stackrel{s}{=} C_i$. Using the notion of

semantic compatibility, we define a *Candidate Service Interface* as a service interface that can accept service request's input objects and provide all outputs that are semantically compatible with the outputs required by the service request. For example (see Table 1), with regard to the service query Q, the interface I_s can be invoked by the service request as the input object `sportsCarA` "is a" `Car` (semantic compatibility). At the same time, I_s can provide all the outputs required in Q since two output parameters are exactly matched. Therefore, I_s is a candidate service interface for Q.

Step 3. Correlation Matching. The system searches a set of service interfaces that can accept the input object from service request and be correlated to provide expected output for the service request. It is worth noting that the type-based service mediation only performs step 1. Most of the semantic service mediations perform semantic matching which is step 2. In our semantic service mediation, we also consider correlation matchings, which are unique to our semantic service mediation. In this paper, we assume that both service consumers and providers use the same ontology for a domain. If a consumers and providers use different ontologies for a domain, then a common ontology can be created. Detailed discussion on creating a common ontology is outside the scope of the paper. Therefore, by engineering ontologies, our system allows different services to exchange information using their native information format to define interfaces and request. The cost of engineering ontologies is much less than that of developing object adaptors for value-based service mediations as ontologies are declaratively defined. Further, ontologies are reusable, while developing object adaptors requires case by case programming efforts.

4 Correlation-Based Service Composition

Obviously, multiple service interfaces can be correlated to one if they have share some input parameters and different output parameters. For example, two service interfaces I_1 and I_2 in $\mathbb{I}_k$ (see Table 1) can be correlated as they both have the field `Car` as the input parameter. Therefore, when the service mediator performs the correlation matching, in order to compose a service interface that can provide all the required outputs for the service request, it first searches a *correlation service interface set*, i.e., a set of service interfaces that are correlatable by a key input parameter that is specified by the service request and can provide all the outputs required by the service request. The formal definition of correlation interface set is shown as follows.

Definition 4 (***Correlation Interface Set***). $\mathbb{I}$ ($\mathbb{I} = \{I_1, I_2, ..., I_n\}$) is a set of service interfaces, $\mathbb{C}_{Pin_i}$ is the class set consists of all the input parameter classes in interface I_i; $\mathbb{C}_{Pout_i}$ is the class set that consists of output parameter classes in I_i; Q is the service request where $\mathbb{C}_{Oin}$ is the class set consists of the input object classes, $\mathbb{C}_{Eout}$ is the class set that consists of expected output parameter classes; o_k is an input object for correlation key. $\mathbb{I}$ is a ***Correlation Service Interface Set*** of Q iff:

1. $\forall C \in \mathbb{C}_{\mathbb{P}in}$, $C \in_s \mathbb{C}_{Oin}$, where $\mathbb{C}_{\mathbb{P}in}$($\mathbb{C}_{\mathbb{P}in} = \cup_{i=1}^{n} \mathbb{C}_{Pin_i}$) is union of all the input parameter class sets in $\mathbb{I}$;
2. $\forall C \in \mathbb{C}_{Eout}$, $C \in_s \mathbb{C}_{\mathbb{P}out}$, where $\mathbb{C}_{\mathbb{P}out}$($\mathbb{C}_{\mathbb{P}out} = \cup_{i=1}^{n} \mathbb{C}_{Pout_i}$) is union of all the output parameter class sets in $\mathbb{I}$;

3. $\forall \mathbb{C}_{Pin_i}, \exists C_k^{'}, C_k \stackrel{s}{=} C_k^{'}$, where C_k is class for key object o_k;
4. $\forall \mathbb{C}_{Pout_i}, \exists C, C \in (\mathbb{C}_{Pout_i} - (\cup_{j=1}^{i-1} \mathbb{C}_{Pout_j} \bigcup \cup_{j=i+1}^{n} \mathbb{C}_{Pout_j}))$ and $C \in_s \mathbb{C}_{Eout}$.

□

In this definition, four conditions need to be satisfied when correlating a set of service interfaces to fulfill a service request: condition 1 indicates any outputs required by the service request can be provided; condition 2 indicates the service request can provide all the required input for each interface in the set; condition 3 implies all the interfaces have the key field as an input parameter, therefore, are correlatable; condition 4 evinces any interfaces in the set contributes at least one unique output. It should be noted that both condition 1 and 2 are necessary condition of the definition, while condition 3 and 4 are the sufficient conditions for the definition. Using above example, the aggregation of I_1 and I_2 provides all the required outputs for the service request, which satisfy condition 1; and their input can be provided by the service request, which satisfies condition 2. These two interfaces have the input parameter `Car` and `Car` is ancestor of `SportsCar`, the key class in service request Q, which satisfies condition 3. Also, I_1 (resp. I_2) provides unique output `CarInsurance` (resp. `CarFinance`), which satisfied condition 4. Therefore, I_1 and I_2 compose a correlation service interface set for the service request.

5 Related Work

Service mediation is a very active area of research and development. In this section, we first review some work in area of service discovery (matching), and then we look at some service composition prototypes.

Service discovery and matching is one of the cornerstones for service mediation. Current Web service infrastructure have limitation on providing flexibility of choose selection criteria along multiple dimensions. For instance, UDDI provides limited search facilities that allows only keyword-based search of services. To overcome this limitation, semantic technology [1] is used to support multiple dimensions searching criterions for services. In [1], a flexible matchmaking between service description and request by adopting Description Logics (DLs). However, most of existing semantic solutions focus on one-to-one matchings. In our service mediation, semantic information in service descriptions and request enables one-to-multiple service matchings, which initiates an other type of automatic service composition.

It should be noted that our correlation-based service composition is different from existing industrial and academic service composition framework. The industrial solution typically does not provide explicit goals of the composition and does not describe the pre- and post-conditions of individual services. A service is viewed as a remote procedure call.A service composition is quite often specified as a process model (e.g. BPEL4WS [2]) though a richer process specification is needed. The composition itself is mostly done manually by IT specialists in an ad-hoc manner. Our approach, using a semantic ontology and a correlation based composition, enables us to construct a composed service based on the semantics of service interfaces, without much programming efforts.

6 Conclusion

In this paper, we propose a novel semantic service mediation, which is another step forward in the development of current service mediation systems. We introduce semantics to understand the interface of service. Our system not only considers single service interface for a service request, but also automatically correlates multiple interfaces for a service request when there is not exactly matched interface. Unlike knowledge-based or process-based solution, the service correlation in our system is transparent to developers. We argue that the proposed service mediation is essential to enable cooperative service interactions in service-oriented computing. Our future work includes optimization of semantic service matching and correlation, and a scalability and reliability study of the system.

References

1. B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani. On automating web services discovery. *The VLDB Journal*, 14(1):84–96, 2005.
2. Business Process Execution Language for Web Services, Version 1.0, 2000. http://www-106.ibm.com/developerworks/library/ws-bpel/.
3. IBM Integrated Ontology Development Toolkit, 2006. http://www.alphaworks.ibm.com/tech/semanticstk.
4. OWL, 2006. http://www.w3.org/TR/owl-ref/.
5. M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *First International Semantic Web Conference*, 2002.
6. Universal Description, Discovery and Integration of Business for the Web, 2005. http://www.uddi.org.
7. L. Zeng, B. Benatallah, H. Lei, A. Ngu, D. Flaxer, and H. Chang. Flexible Composition of Enterprise Web Services. *Electronic Markets - The International Journal of Electronic Commerce and Business Media*, 2003.
8. L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.