Multimed Tools Appl (2006) 30: 149–164
DOI 10.1007/s11042-006-0022-6

An open framework supporting multimedia web services

Jia Zhang · Jen-Yao Chung

Published online: 5 August 2006

Abstract With the rapid emergence of Web services, more and more Web services are published on the Internet as resources for Web application development. There may exist some relationships among different Web services, such as exact match, plug-in match, and irrelevant. In this paper, we discuss a set of requirements related to multimedia Web services, and propose a three-tier framework to establish an open environment supporting multimedia Web services, while partially implementing the requirements. This paper focuses on the design of the service broker tier that is essential for future Web services-oriented system design and integration and enabling Web services more transparent, interoperable, and fault-tolerate.

Keywords Multimedia Web services · Service broker · Three-tier open environment · Requirements

1 Introduction

Web services are broadly regarded as self-contained, self-describing, modular applications that can be published, located, and invoked across the Internet [11]. This emerging paradigm opens a new cost-effective way of engineering software to quickly develop and deploy Web applications by dynamically integrating other independently published Web services as components to conduct new business transactions. As changing the Internet from a repository of data to a repository of services, the model of Web services has been obtaining significant momentum in both academia and industry in recent years. However, since the Web services components are actually integrated at run time through the Internet, one essential problem arising is how to guarantee that a Web service can be obtained dynamically with transparency and fault tolerance.

Roy and Ramanujan [14] summarize a typical architectural model for Web services that shows a triangular relationship among three components: service providers, service brokers, and service requesters. As illustrated in figure 1, service providers publish Web services to

J. Zhang (✉)
Department of Computer Science, Northern Illinois University, DeKalb, IL 60115, USA
e-mail: jiazhang@cs.niu.edu

J.-Y. Chung
IBM T.J. Watson Research, Yorktown Heights, New York, NY 10598, USA
e-mail: jychung@us.ibm.com

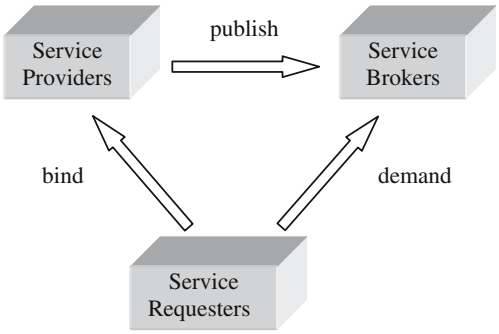

Fig. 1 Typical Web services architecture

service brokers; service requesters demand services from service brokers; and then service requesters directly bind to particular service providers. This framework works well at the early stage of Web services realm, when different Web services do not interoperate with each other. As people start to utilize published Web services as components to construct larger business process, problems arise. Figure 2 shows a typical example that contains multiple Web services using the architecture in figure 1. Suppose one Web application needs to integrate three published Web services, each from different service providers 1, 2, 3, respectively. For simplicity, we assume that these three service providers publish their services on the same service broker. After the service requester obtains the locations of these three services from the service broker, it needs to invoke the three Web services from individual service providers. Therefore, the service requester needs to be aware of the details of the service providers, such as their locations, and even the port numbers of the desired services. If an unexpected accident occurs on the Web, say, the service provider 1 crashes, the service requester has to re-access the service broker for a substitute, and re-establish a connection to the new service provider. As a result, the service requester has to end up handling the invocations and error handlings of every Web service needed, which is obviously neither efficient nor effective.

Furthermore, when Web services contain multimedia elements, in which case called multimedia Web services, things are even exasperated. First, only the services that satisfy the multimedia quality of service (QoS) at the invocation time should be taken into consideration, such as latency tolerance, real time synchronization, network bandwidth, etc.

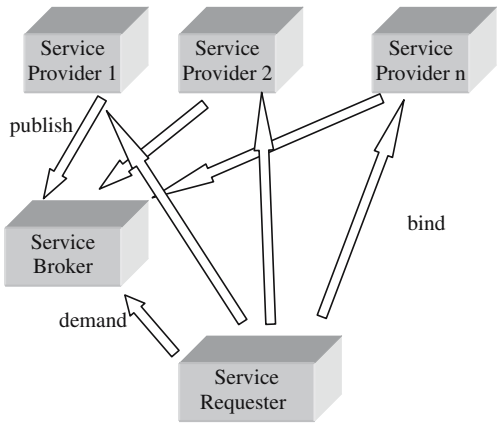

Fig. 2 More sophisticated situation

It should be noted that in this paper, the term QoS refers to multimedia-related quality features. Second, if for a service requester, different media contents that need to be synchronized will be provided by different Web services, dynamic synchronization mechanisms are imperative.

Our goal in this research aims at establishing an open framework to support multimedia Web services. To realize our goal, we first discuss a set of requirements related to multimedia Web services, and then propose a three-tier framework to establish an open environment supporting multimedia Web services, while partially implementing the requirements. This paper focuses on the design of the service broker tier that is essential for (1) future Web services-oriented system publication, selection, and invocation, and (2) enabling multimedia Web services more transparent, interoperable, and fault-tolerate. Throughout this paper, when we use the term *Web services* or *service*, we refer to *Multimedia Web services*. The remainder of the paper is organized as follows. In Section 2 we present the requirements of supporting multimedia Web services and mechanisms needed. In Section 3 we discuss the related work. In Section 4 we define some basic concepts. In Section 5 we propose a three-tier framework supporting multimedia Web services. In Section 6 we describe the prototype implementation of the framework. In Section 7 we present self-assessment. In Section 8 we draw conclusions.

2 Problem domain definition

As the first step, we discuss the requirements of supporting multimedia Web services. Eight basic requirements are identified as follows: (1) transparency, (2) management of service relationships, (3) dynamic selection and composition, (4) coordination, (5) multimedia quality of service, (6) fault tolerance, (7) distribution of Web services registration, and (8) criteria to choose services. Table 1 summarizes these requirements and the associated mechanisms that we believe to be capable of fulfilling each requirement. Since our research focuses on the publication, selection, and invocation of multimedia Web services, in this paper we omit the discussions for other possible requirements, such as resource localization, multicast support, support for continuous media, real time synchronization, security, etc.

Table 1 Requirements and mechanisms

Requirements	Mechanisms
Transparency of locating and invoking Web services	Encapsulation of the details of location and invocation
Management of relationships between published Web services	Definition of the service relationships
Dynamic selection and composition of Web services	Dynamic selection and dynamic binding
Coordination of Web services	Management of relationships and the information exchange
QoS awareness	Support of the expression of QoS parameters
Fault tolerance	Dynamic selection and dynamic binding
Distribution of Web service registration	Trading between service brokers, caching
Criteria to choose Web services	Definition of the service relationships, support of the expression of QoS parameters, popularity tracking

The first requirement is the transparency of locating and invoking Web services. Similar to the network transparency [13], Web service transparency means that Web services behave in the same way independent of their distributed locations and execution environments. In other words, the distinctions between the locations of Web services should be invisible to service requesters; and a Web service running on a remote site should behave the same way when running on a local site. To achieve the transparency, mechanisms need to be provided to encapsulate the details of the locations and invocations of Web services.

The second requirement is the management of the relationships between published Web services. A large amount of Web services have already been published on the Internet, and the number and the types of Web services grow rapidly [5]. How to choose an appropriate Web service and how to choose a substitute when one Web service is unavailable are of paramount importance. Therefore, mechanisms need to be provided to define the relationships between Web services, such as exact matching, substitutable matching, etc.

The third requirement is the dynamic selection and composition of Web services into a new business transaction. In a Web services-oriented environment, the number of available services constantly changes, since at any time new Web services may be added into the Internet and old ones may be removed [5]. In addition, due to the unpredictable features of Internet, some pre-selected Web services may become temporarily unavailable at some time; therefore, other compatible Web services should be selected as replacements. As a result, applications based upon Web services should be able to choose and compose Web services at run time. Furthermore, since Web services will be dynamically selected, static binding at compiling time and linking time is not practical. Therefore, dynamic binding needs to be supported.

The fourth requirement is the coordination of Web services. Due to the fact that an enterprise application may request supports from multiple Web services, these services need to be coordinated harmoniously and exchange information effectively. To achieve this goal, mechanisms need to be provided to manage the relationships among Web services and handle the information exchange among them.

The fifth requirement is the multimedia quality of service (QoS) awareness. QoS requirements, one of the essential features of multimedia services [6], guarantee the timeliness of multimedia transmissions. Common QoS parameters include: throughput, end-to-end delay (latency), delay variance (jitter), etc. Therefore, the selection of multimedia Web services should not only be based upon availability, but also upon the QoS characteristics. In order to select the Web services that fulfill the QoS requirements, mechanisms need to be provided to support the expression of QoS parameters, so that Web services can be selected based upon their QoS values if so desired. In addition, a Web service may become overloaded at some point and stop responding in a timely fashion, which will violate QoS requirements. In this paper we do not discuss the mechanism to ensure the QoS through the Internet transmission.

The sixth requirement is fault tolerance. The fault tolerance here refers to the ability of a Web services-oriented system to respond gracefully to an unexpected failure of a Web service component. Let us consider an application that is composed of several Web services and is executed the second time. From its first time of execution, the set of Web services used will be cached. Since each Web service will be invoked remotely from its resident site at the time of the invocation, it is possible that one Web service crashes without warning after the first invocation. The system then needs to be able to make some special arrangements to find a new Web service to replace the failed one, so that a service requester will still obtain the whole application even through a certain Web service is crashed. Dynamic selection and binding of Web services can be the mechanism to achieve this goal.

It should be noted that for simplicity, in this paper we do not consider fault tolerance due to the issue of mid-service failure.

The seventh requirement is the distribution of Web services registration. Web service providers always register to some service brokers. As more and more Web services are published on the Internet, it is infeasible to have only one central service broker that handles the entire pool of published Web services. As a result, there will be many service brokers on the Web, each managing some Web services. These service brokers can be further differentiated and established based upon different business purposes. For example, one service broker may be constructed to hold registrations of on-line computer science courses. Furthermore, one such service broker may be set in North America, while another one set in Asia, yet another one set in Europe. When a Web service is requested, some strategy might be adopted to select a service broker, e.g., the closest service broker will be first checked. If the expected Web service is not found, the service broker should automatically contact with other service brokers for the appropriate service. If the service is found elsewhere, the original service broker should duplicate the service information to its local storage. Mechanisms should be provided to support the trading between service brokers described here.

The eighth requirement is the criteria to choose appropriate Web services. As more and more Web services are published on the Internet, the selection pool of services serving for similar purposes is becoming larger and larger. How to choose the most appropriate Web services becomes challenging. To address this issue, we believe that at least three criteria need to be considered: (1) the relationships among Web services have to be defined; (2) the QoS parameters should be expressed; and (3) the popularity of Web services should be taken into consideration. The popularity of a Web service here refers to the constancy of access of a Web service in the historical standpoint.

3 Related work

Remote Procedure Call (RPC) is a powerful mechanism in distributed computing, which enables software to make procedure calls over the Internet onto another procedure running on distributed machines. Providing a level of abstraction above the underlying message stream, RPC facilitates a client/server model, in which a server defines a set of services in the format of procedures that can be called by remote clients. XML–RPC [12], which is a popular protocol that uses eXtensible Markup Language (XML) [2] as encoding over Hypertext Transfer Protocol (HTTP) [4] as transport to realize remote procedure calls [10]. XML–RPC differs from the traditional RPC in several significant ways: (1) XML–RPC utilizes the standard XML encoding strategy so that systems can be loosely coupled and highly interoperable; (2) the issue of argument marshaling existing with the traditional RPC is resolved because all data are encoded as text before transmission; and (3) XML–RPC is integrated with existing transportation protocols as HTTP [1]. Apache XML–RPC [7] is a Java implementation of XML–RPC. Although XML–RPC is simple to understand and use, its goal of simplicity decides that it cannot handle complex data types. Simple Object Access Protocol (SOAP) [8], on the other hand, is a more comprehensive and powerful transportation protocol, which can handle complex data types such as user-defined data types, and has the ability to have each message define its specific processing control and recipient. Becoming *ad hoc* standard in the Web services field, the SOAP specification defines a convention to represent RPC calls and responses. Therefore, SOAP covers XML–RPC and

provides more power of supporting Web services-oriented system. As a result, in our research, we decide to adopt SOAP RPC to access remote Web services. Meanwhile, our previous work enhances SOAP to improve the ability and flexibility of the SOAP protocol to serve multimedia Web services, by supporting batch facility and carrying on QoS requirements [18]. Consequently, we utilize our enhanced SOAP to transfer request messages.

Researchers, especially those from the field of Web services discovery, have been interested in identifying QoS features as requirements of locating Web services. In UX [3], service requesters are prompted to give QoS feedbacks, so that the system can generate summaries for invoked services. Three QoS parameters are suggested: (1) response time, (2) cost, and (3) reliability. Vinoski [17] also summarizes five QoS parameters: (1) latency as the average time for an operation to return results after its invocation, (2) fees as the money needed to be paid to invoke operation, (3) availability as the probability that the Web service is present and ready to be invoked, (4) accessibility as the degree of being capable of serving a request, and (5) reliability as the degree of being capable of maintaining the service and service quality. In our work, we choose to adopt several multimedia-related QoS parameters: (1) response time, (2) reliability, (3) availability, and (4) accessibility.

Meanwhile, a powerful language to formally and precisely define a Web service is of paramount importance. Web service description language (WSDL) [9] from W3C is becoming the ad hoc standard for Web services publication. However, WSDL can only specify limited information of a Web service as the function names and limited input and output information [5]. Gao et al. [5] propose a Web service capability description language (SCDL), which is built upon the method of abstract finite-state machine, to describe, advertise, request, and match Web services capabilities precisely. A Web service is defined as a frame of structure as: name, ontological description, type, input variables, output variables, pre-conditions, and post-conditions. SCDL defines the four types of atomic Web service capability matches: exact match, plug-in match, relaxed match, and not relevant. The authors also formally define the plug-in match utilizing the formal logic methods. The paper provides a theoretical basis to define Web service capability matching. However, the paper does not provide any information about the implementation of the SCDL language; its previous version SDL [16] is still at early development stages [15]. Therefore, the usage of SCDL in Web services applications is still unclear.

4 Basic definitions

To address the issues discussed above, a three-tier framework supporting multimedia Web services is proposed. To facilitate our discussions of the framework, however, we need to define some basic concepts first.

4.1 Definition 1: Web service

In this paper, each Web service is defined as a six-tuple (hostId, ontoDes, Sigs, Pre, Post, QoS), where:

- hostId: is the unique identifier of the hosting server machine of the Web service;
- ontoDes: is the ontological description. This element defines the concept of the context and its meaning description [5].

Sigs: is the set of RPC methods exposed by the service:

Sigs ::= {M1 V M2, . . . ,V Mi . . . }, i≥1
Each RPC method can be defined as follows:

M ::= (N, i_1, i_2, . . . , i_m, o_1, o_2, . . . o_n) where:
N: the name of the method;
i_1, i_2, . . . , i_m: the list of the types of the input parameters;
o_1, o_2, . . . o_n: the list of the types of output parameters;

- Pre: is the pre-condition of the Web service;
- Post: is the post-condition of the Web service;
- QoS: is the QoS feature of the Web service.

4.2 Definition 2: signature match

Considering two RPC methods A and B exposed by two Web services, method A is regarded as signature matching to method B if:

1. The input parameters of A are super types of those of B;
2. The output parameters of A are subtypes of those of B.

Example: For simplicity, we utilize Java method signature for our discussion. Let us consider two mathematical methods that calculate a square root of a number. The first method takes an integer type as the input parameter, and outputs a real type. The signature is as follows:

```
double squareRoot (int inputNumber);
```

The second method takes a real type as the input parameter, and outputs a real type. The signature is as follows:

```
double squareRoot (float inputNumber);
```

Since the input parameter of the second method is of type real, which is a super type of the type integer of the first method, and the output parameter of the second method is the same as that of the first method, we consider that the second method is signature matching with the first method.

4.3 Definition 3: plug-in match Web service

Plug-in match defines a substitute relationship between Web services. If Web service X is a plug-in match service of Web service Y, it means that Web service X can be plugged into the place as a substitute where Web service Y is to be used, but not vice versa. A Web service X is a plug-in match service of Web service Y, if:

1. The method set defined by X is a superset of that defined by Y;
2. For each mutual method between X and Y, the signature of the method signature of X matches that of Y.
3. The specification of X semantically matches the specification of Y. In other words, (pre-X => pre-Y) ^ (post-Y => post-X) [Gao2].
4. For every QoS requirement defined in Y, X fulfills the same QoS requirement with super type (i.e., for every QoS specification, X satisfies with stronger features).

4.4 Definition 4: exact match Web service

Exact match Web service defines that two Web services are potentially interchangeable. Two Web services, say X and Y, are considered to be exact match if X is plug-in match with Y and Y is plug-in match with X.

4.5 Definition 5: related Web services

Related Web services defines that two Web services are either exact match or plug-in match. Otherwise, two Web services are considered irrelevant services.

4.6 Definition 6: service domain

A service domain in this paper is a conceptual term for the purpose of management of Web services. A service domain is defined based upon the distribution of Web services registrations. All service providers who register on a service broker form a service domain together with the service broker. As new services register onto the service broker, or some old services remove from the service broker, the boundary of the service domain alters accordingly. Therefore, the concept of a service domain represents a set of published Web services.

5 Three-tier framework supporting multimedia Web services

Based upon our previous discussions, in this section, we present a three-tier framework for multimedia Web services, which aims to support the integration of the mechanisms we propose to fulfill the requirements. As illustrated in figure 3, there are three layers in the framework: service providers, service brokers, and service requesters. Multiple service providers register onto the same service broker; multiple service requesters access the same service broker. It should be noted that our framework differs from the normal Web services architectural model described in Section 1 in the following two ways. First, in the normal architectural model, as we discussed in Section 1, when a service requester asks for a Web service, it sends a request to the service broker. After the service broker finds the expected service, it will return the service provider's information to the service requester. Then the service requester will connect to the specific service provider for the service by itself. In our model, on the other hand, the service broker serves as the middle tier between service providers and service requesters. Service requesters do not connect to service providers directly; instead, they indirectly communicate with each other via the service broker. Second, in our framework, the service broker becomes much "fatter" than it is in the normal model. It not only serve service registration and management, but also serve dynamic service selection and binding, caching, service trading, etc. In other words, in our model, the service broker tier integrates many intelligence and management functionalities; therefore, the service providers and the service requesters become much thinner components in the model.

As shown in figure 3, the service broker is wrapped by a XML layer. This XML layer can be implemented by different XML-based technologies, such as SOAP, UDDI, and WSDL. For brevity, this paper will not discuss the related technologies; therefore, we use *XML layer* to merely represent that the communications between three tiers are all based upon the XML technology. As illustrated in figure 3, the internal structure of a service

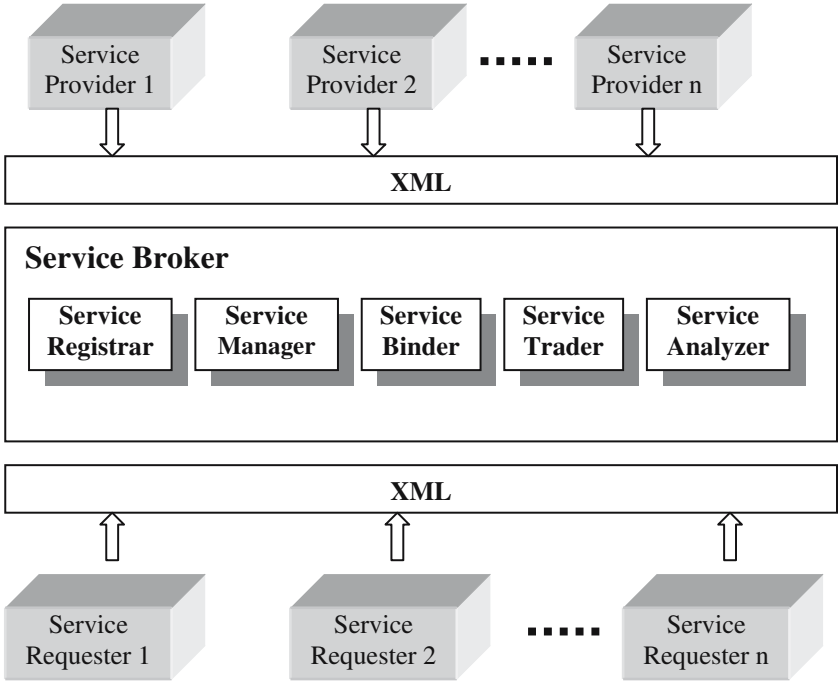

Fig. 3 Three-tier framework supporting multimedia Web services

broker contains the following five functional components: (1) service registrar, (2) service manager, (3) service binder, (4) service trader, and (5) service analyzer.

The service registrar handles service registrations for service providers and the service trader component, which will be discussed later. Service providers register new Web services onto the service registrar, or remove old services from it. Commonly, the service registrar maintains one repository of registered Web services, and also provides an operation engine over the repository (e.g., adding/removing a service entry), and query functionality for the service binder, which will be discussed later. Through the service registrar, the details of Web services are hidden from service requesters. The service trader component can also register Web services into the service registrar, which scenario will be discussed later.

The service manager is meant to manage the registered Web services to realize the dynamic selection and binding of QoS-aware Web services with transparency and fault tolerance. Managing the relationships between registered Web services, the service manager selects an appropriate service from the service pool at run time, based upon the functionality and QoS requirements. Then the service binder will try to bind to the selected service provider. If the chosen service is not available, or cannot satisfy the QoS requirements at the moment, the service manager will look into its service pool again for a new service. If there are no related services available at the moment, the service manager will notify the service requester to try at a later time. The service manager normally maintains one repository of registered services grouped with relationships, e.g., exact matching or plug-in matching.

The service binder dedicates in providing dynamic binding services for service requesters. Either a service requester or the service manager can invoke the service binder

for a service. On receiving requests, the service binder will query the registered service repository for the proper binding properties, such as the service host machine identifier and the service name. In other words, the service binder is in charge of establishing the connections for service requesters. If the service binder cannot set up the connection (e.g., the service host machine is overloaded), the service binder will notify the service manager for a substitute Web service, and try to build the connection again.

The service trader manages the trading facility among different service domains. A service requester sends a request to a service broker, maybe because it is a member of the service broker, or because the specific service broker is in its local area. It is possible that a requested Web service cannot be found on one service trader (i.e., the service requested is not in the service domain). Then the service trader needs to forward the request to other service brokers, or service domains, for the particular Web service. The QoS requirements associated with the request should be sent as well. Therefore, the service trader needs to maintain a pool of other service brokers. If a service is found from another service broker, the original service trader will register the Web service onto the service registrar component in the same service broker. When a service registration is copied over, the associated popularity, which will be discussed below, will be copied as well. Therefore, this duplicate copy of service registration can serve future requests in the original service domain. Meanwhile, when a service trader receives requests from another service broker, it will check the service manager component for the requested service.

The service analyzer is a utility component, which provides statistical analysis of the popularity of registered Web services in a service domain. For example, whenever a binding between the service binder and a service provider is successfully set up, the counter associated with the specific Web service will be increased. The higher the counter is, the higher popularity the service provider will be. The popularity will be one of the essential criteria for the service manager to select appropriate Web service, when multiple registered Web services provide similar functionality and QoS parameters.

6 Implementation

We have implemented a prototype system based upon our three-tier framework, whose structure is illustrated by figure 4. For brevity, in our system, all service providers expose their Web services with RPC interfaces. Here we will focus on the structure of the service broker. We utilize our enhanced SOAP protocol [18] to serve the communication channels between the service broker, service requesters, and service providers, due to its ability to transfer QoS parameters and facilitate multimedia transportation. Since we have not found an ideal description language for Web services publication as we discussed in the previous section, in this prototype system we implement a registrar component with interfaces for service providers to register their Web services with QoS requirements. This is certainly just a temporary solution, but it can help us prove the concept of our framework. Furthermore, this module can be easily upgraded with a description language in the future.

In the service broker, we implemented five main functional modules and three repositories. The five modules are: (1) service registrar, (2) service manager, (3) service binder, (4) service trader, and (5) service analyzer. The functionality of each module follows our framework. The three repositories are: (1) service repository, (2) broker repository, and (3) historical repository. These repositories store service information, other service brokers' information, and historical successful access information, respectively. Figure 4 also shows

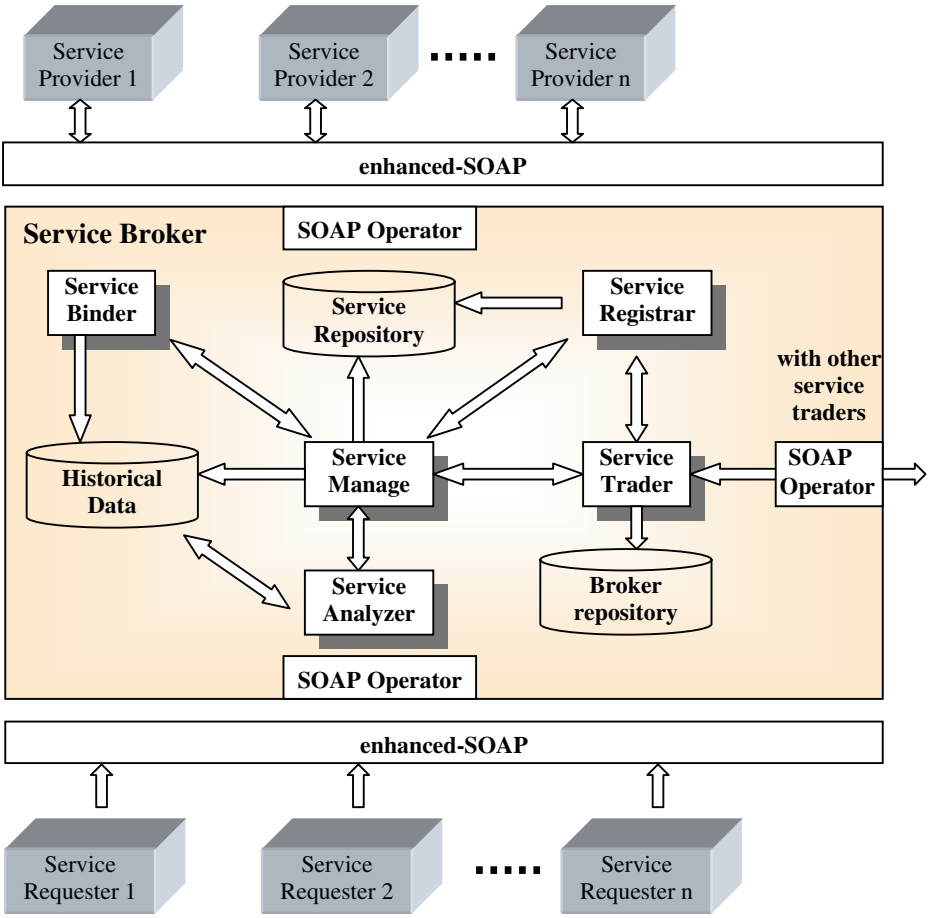

Fig. 4 Implementation of three-tier framework

the relationships among them and the access paths among them. The service broker contains SOAP operators to generate and interpret to and from SOAP messages. Notice that there are three SOAP operators in figure 4. As a matter of fact, only one SOAP operator exists. The reasons of showing multiple SOAP operators are: (1) for displaying purposes for easier painting, and (2) to emphasize that the SOAP translation or generation are necessary at three places. For brevity, in the following descriptions, we omit the steps of translation/generation of SOAP messages, as they are always necessary when the service broker communicates externally.

The service providers publish their services to the service broker through the service registrar module. The registration information follows our definition of a Web service in previous section, which includes functionality, host machine identifier, QoS parameters, etc. The service registrar then stores the service information in the service repository.

When a service requester asks for a specific service, the request should contain the desired functionality of the service and its QoS requirements. The request will be forwarded to the service manager, and the latter one searches from the service repository and historical repository for an appropriate Web service registered. Notice here that the service repository

contains different views and categories based upon the matching relationships between the Web services, as we discussed in the previous section. In our prototype system, we provide two options for the service requester. One is that the service requester give the service broker full right to automatically decide which Web service to choose. The other one is to let the service broker provides candidate services, and leave the service requester to decide which one to invoke. To facilitate our discussion, here we assume to adopt the first option. The service manager will then coordinate with the service analyzer to decide the most appropriate Web service. We will also skip here the algorithm we used to a Web service, and simply assume that the service broker always select the most appropriate Web service, based upon QoS requirements and popularity.

If a service is found successfully from the service repository, the service binder module will try to establish the connection to the host machine containing the chosen Web service. If succeeded, the connection will be passed back to the service requester, and the service binder will record the successful access information into the historical repository. Otherwise, the service manager will repeat the previous action to search for a replacement, and the service binder will continue to bind to the new host machine with the newly selected service. If the service is not found in the service repository, the request will be forwarded to the service trader, and the service trader will search for the broker repository to find other service domains for the expected service. Figure 5 illustrates the concept of service domain. All the service providers that register to a service broker form a service domain, together with the service broker. One service provider may have registrations on multiple service brokers; therefore, the service provider may belong to multiple service domains. As shown in figure 5, service providers 1, 2, . . . , m have registrations on the service broker A, therefore, they shape a service domain 1. Similarly, service providers m, n, . . . , q form another service domain 2, as they all register on the service broker B. The

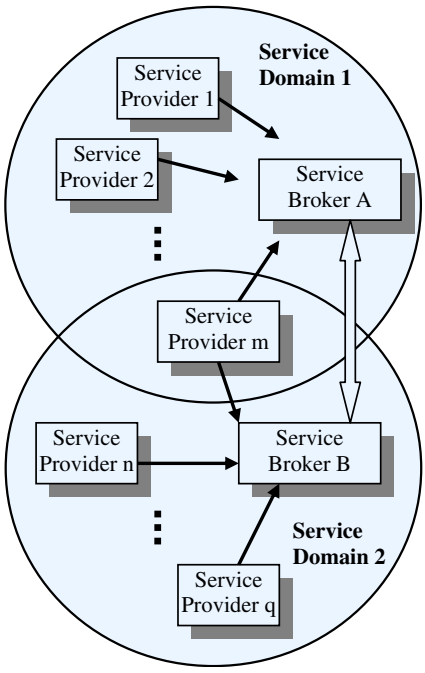

Fig. 5 Management of service domain

service provider m belongs to both service domains 1 and 2. We can notice that the service broker is the representation of the specific service domain. Service brokers from different service domains may interact with each other to share the registered Web services. If one of the associated service brokers finds the desired service in its own service domain, the registration information will be passed back to the original service broker, and the information will be registered to the service repository through the service registrar. As a new Web service is registered, the range of the service domain is correspondingly enlarged.

7 Evaluation

The main evaluation of our work was conducted to examine the effect of our framework against the research issues we discussed in Section 2 problem domain specification. For each issue, we scrutinize which solution our framework proposes, which components of our framework are involved, and check whether the issue has been fully resolved or just partially resolved. The evaluation result is summarized in Table 2. For other issues related to multimedia Web services, we are not discussing in this paper.

- *Transparency of locating and invoking Web services*: The service manager, service binder, and service trader are involved. The service manager selects an appropriate service, and the service binder connects to the chosen service. If the desired service does not register in the service domain, the service trader will be involved to find one in other service domains. In other words, the detailed information about the location and the invocation of a proper service is masked from the service requester. Therefore, this issue is solved by our framework.
- *Management of relationships*: The service manager is involved. The service manager decides the matching relationships between registered Web services, and groups them accordingly to provide different views of the service repository. Consequently, the service relationships are maintained. However, in reality, since we have not found an ideal Web service description language that is powerful enough to describe the characteristics of a multimedia Web service, this issue will remain partially resolved until we find a better solution.

Table 2 Requirements solving results

Requirements	Components involved	Result
Transparency of locating and invoking Web services	Service manager, service binder, service trader	Solved
Management of relationships between published Web services	Service manager	Partly solved
Dynamic selection and composition of Web services	Service manager, service binder, service trader	Partly solved
Coordination of Web services	Service manager, service binder	Partly solved
QoS awareness	Service manager, service binder	Partly solved
Fault tolerance	Service manager, service binder	Solved
Distribution of Web service registrations	Service trader	Solved
Criteria to choose Web services	Service manager, service analyzer, service binder	Solved

- *Dynamic selection and composition*: The service manager, service binder, and service trader are involved. The service manager conducts run-time selection of appropriate Web services based upon their availability and popularity. The service binder helps guarantee the availability of the Web services, and the service trader helps locate services. However, due to the same reason as above, this issue cannot be fully resolved unless a powerful description language appears. In addition, in this paper we do not discuss the issue of service composition.
- *Coordination of Web services*: The service manager is mainly involved. The service manager checks the pre-condition and post-condition of involving Web services to handle the coordination of Web services. However, due to the same reason as above, this issue cannot be fully resolved unless a powerful description language appears.
- *QoS awareness*: The service manager and service binder are mainly involved. The service manager utilizes QoS parameters as a criterion to select services; and the service binder connects to the host machines of the services to check whether the QoS parameters remain the same at the moment. However, due to the same reason as above, this issue cannot be fully resolved unless a powerful description language appears.
- *Fault tolerance*: The service manager and service binder are mainly involved. The service manager and the service binder cooperate to achieve dynamic service selection. When a service is not available, the service manager will research for a substitute. Therefore, this issue is resolved. It should be noted that for simplicity, in this paper we do not consider the fault tolerance due to the issue of mid-service failure.
- *Distribution of Web service registrations*: The service trader is involved. The service trader facilitates Web services registration at different service brokers locally. At run time, service traders can interact with each other to share the Web service registration information. Therefore, the issue is solved.
- *Criteria to choose Web services*: The service manager, service binder, and service analyzer are involved. The service manager conducts run-time selection of appropriate Web services based upon functionality, QoS constraints, availability, and popularity derived from historical records. This combination of criteria is comprehensive to facilitate service selection. Therefore, the issue is solved.

Based upon our evaluation result, we conclude that our framework to large degree solves several essential issues related to multimedia Web services.

8 Conclusions and future work

This paper discusses the issues related to multimedia Web services-oriented environment, and proposes a three-tier framework to achieve transparent, dynamic, and multimedia-enabled Web services with fault tolerance. We also discuss our implementation of the framework. This research work leads to establishing an open environment supporting multimedia Web services.

Our future work includes the following directions. First, we will explore a Web service description language in order to support our requirements. Second, we will investigate the notification services between service domains. Third, we need to examine the effect of the security issue in our framework.

References

1. Allman M (2003, March) An evaluation of XML–RPC. ACM SIGMETRICS Perform Eval Rev 30 (4):2–11
2. Bray T, Paoli J, Sperberg-McQueen CM, Maker E (2000) Extensible Markup Language (XML) 1.0, 2nd Ed. W3C. http://www.w3.org/TR/2000/REC-xml-20001006.pdf
3. Chen Z, Chia L-T, Silverajan B, Lee B-S (2003, June 23–26) UX—an architecture providing QoS-aware and federated support for UDDI. In: Proc. of the Int. Conf. on Web Services (ICWS'03), Las Vegas, Nevada, pp 171–176
4. Fielding R, Gettys J, Mogul J, Frystyk H, Berners-Lee T (1997) Hypertext Transfer Protocol—HTTP/1.1, IETF. http://www.ietf.org/rfc/rfc2068.txt
5. Gao X, Yang J, Papazoglou MP (2002, December 11–13) The capability matching of web services. In: Proc. of the IEEE Int. Symposium on Multimedia Software Engineering (MSE'02), Newport Beach, California, pp 56–63
6. Hong DW, Hong CS (2003, March/April) A QoS management framework for distributed multimedia systems. Int J Netw Manage 13(2):115–127
7. http://ws.apache.org/xmlrpc
8. http://www.w3.org/TR/SOAP
9. http://www.w3.org/TR/wsdl
10. http://www.xmlrpc.com
11. IBM Web Services tutorial. http://www-106.ibm.com/developerworks/Webservices
12. Laurent S, Johnston J, Dumbill E (2001) Programming web services with XML–RPC, O'Reilly
13. Reed DP, Saltzer JH, Clark DD (1998, May/June) Comment on active networking and end-to-end arguments. IEEE Netw 12(3):69–71
14. Roy J, Ramanujan A (2001, November/December) Understanding web services. IEEE IT Professional 3(6):69–73
15. Tsalgatidou A, Pilioura T (2002) An overview of standards and related technology in web services. Distributed and Parallel Databases 12:135–162
16. van den Heuvel WJ, Yang J, Papazoglou M (2001, September 5–7) Service representation, discovery and composition for e-marketplaces. In: Proc. of the 6th Int. Conf. on Cooperative Information Systems (COOPIS'01), Trento, Italy
17. Vinoski S (2003, January/February) Service discovery 101. IEEE Internet Computing 7(1):69–71
18. Zhang J, Chung J-Y (2002, December 11–13) A SOAP-oriented component-based framework supporting device-independent multimedia web services. In: Proc. of IEEE 4th Int. Symposium on Multimedia Software Engineering (MSE'02), Newport Beach, California, pp 40–47

Jia Zhang, Ph.D., is an Assistant Professor of Department of Computer Science at Northern Illinois University. She is currently with BEA Systems Inc.; and also a Guest Scientist of National Institute of Standards and Technology (NIST). Her current research interests center around software trustworthiness in the domain of Web services, with a focus on reliability, integrity, security, and interoperability. Zhang has published over 60 technical papers in journals, book chapters, and conference proceedings. She also has seven years of industrial experience as software technical lead in Web application development. Zhang is an Associate Editor of the International Journal of Web Services Research (JWSR). She is now serving as Program Vice Chair of IEEE International Conference on Web Services (ICWS 2006). Zhang received a Ph. D. in computer science from University of Illinois at Chicago in 2000. She is a member of the IEEE and ACM. Contact her at jiazhang@cs.niu.edu.

Dr. **Jen-Yao Chung** received the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign. Currently, he is the senior manager for Engineering & Technology Services Innovation, where he was responsible for identifying and creating emergent solutions. He was Chief Technology Officer for IBM Global Electronics Industry. Before that, he was senior manager of the electronic commerce and supply chain department, and program director for the IBM Institute for Advanced Commerce Technology office. Dr. Chung is the co-founder and co-chair of IEEE technical committee on e-Commerce (TCEC). He has served as general chair and program chair for many international conferences, most recently he served as the steering committee chair for the IEEE International Conference on e-Commerce Technology (CEC05) and general chair for the IEEE International Conference on e-Business Engineering (ICEBE05). He has authored or coauthored over 150 technical papers in published journals or conference proceedings. He is a senior member of the IEEE and a member of ACM.