A Service-Oriented Architecture for Design and Development of Middleware

Yih-Cheng Lee*
Dept. of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
m9321054@em93.ndhu.edu.tw

Chi-Ming Ma
Dept. of Information Management, Dahan Institute of Technology, Hualien, Taiwan
cmma@ms01.dahan.edu.tw

Shih-Chien Chou
Dept. of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
scchou@mail.ndhu.edu.tw

Abstract

Middleware is an intermediate layer in software architecture, which helps application developers write program codes without understanding the complexities behind the middleware. In addition to this, middleware also provides transparencies, for example, location transparency and logic transparency. In this paper, we design and develop a middleware in the way of service-oriented architecture (SOA) and it provides a new transparency -- 'Service Transparency'. We will explain what this middleware concerns, how it is developed, and what services it provides.

Keyword: Middleware Design and Development, Service-Oriented Architecture, Web Services, XML, Interoperability.

* Corresponding author

1. Introduction

Middleware is a special software that acts as a conversion or translation layer to consolidate or integrate data in the environment. Today, we have many middleware with different purposes running on different platforms and produced by different vendors. For example, Messaging middleware provides a common interface and transport between applications. It helps 'Application A' pass necessary data to 'Application B' easily. Besides the messaging middleware may contain business logic that routes the data to appropriate destinations and re-organize the data in different data format as well. E-mail system is a good example of messaging middleware. Besides Messaging middleware, today we also use many different middleware for our applications. For example, database connection provider is a middleware that helps applications connect to database and get or set data easily. ODBC [5] is a famous and world-wide use middleware for database connectivity.

Today, a software start-up would like to create their software or applications for reusing in the future. This not only helps software projects being developed quickly but also accumulates business knowledge and associated programming codes of business logic. In view of this, we design and develop a service-oriented middleware which provides not only the transparencies as we mentioned before but also ideas of service-oriented architecture.

2. Services-Oriented Design

Services are what we connect together by the standard implementation. A service is an endpoint of a connection. For example, Auto Teller Machines are setup by banks and located in many different locations in order to provide the convenient financial services to the customers. People can deposit and withdraw money in front of that machine. ATM is the endpoint of connection to people and the connection connects to all necessary operations needed to complete deposit and withdraw commands.

A service-oriented architecture is a collection of services. These services communicate with others. The communication can involve either simple data passing or it could involve two or more services coordinating some activities. Service-oriented architectures provide a promising way to address problems related to the integrations of heterogeneous applications in a distributed environment [1].

Before designing and developing the service-oriented middleware, we have to decide that what technology

Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)
0-7695-2465-6/05 $20.00 © 2005 IEEE

is the best choice for implement the service-oriented middleware. After comparisons and considerations, XML web services is the best choice under our consideration because of some reasons as follows,

Reason 1: XML is supported by W3C [2]. It is well-supported by famous application platform, e.g. Sun Java platform [3] and Microsoft .Net Framework [4]. XML web services use XML as the data format transporting among applications.

Reason 2: XML web services use HTTP as the transportation protocol. Since HTTP is a very popular protocol in most organizations, it can be used to the transportation protocol of heterogeneous applications in a distributed environment.

Reason 3: XML web services provide a loosely-coupled way to interoperate with other services.

Reason 4: The interfaces, called *portTypes* in WSDL [6], which are collections of operations, each of which describes a particular pattern of message exchanges [1].

According to these reasons, we choose to implement the service-oriented middleware by XML web services technology.

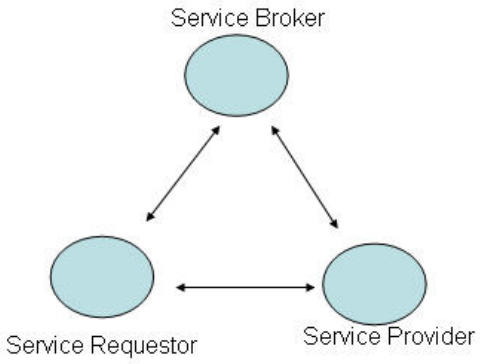

Figure 1. Basic SOA conceptual model

Figure 1 states the conceptual model of service-oriented architecture. Service Provider provides services to the public. It announces and registers the services it provided to Service Broker. Therefore, we can see that the Service Broker ultimately becomes a service repository that keeps a large set of services' records. Service Requestor, also known as Service Consumer, sends a request to Service Broker and searches a service it needs by a keyword or service name in the service repository. At last, Service Requestor gets the address of the service and connects to the Service Provider.

A simple service may be provided by a Service Provider. However, a complex service may be provided by more than one Service Provider. In this case, Service Requestor has no idea how to orchestrate the services from Service Providers to compose the correct sequence of a service. For example, when people buy products from Internet E-commerce websites, all websites provide the procedure of checkout operation. All people have to do is follow the checkout procedure. However, checkout procedure is not a simple procedure because it contains the shipping process and the payment validation process. When people start the checkout operation, he/she probably is led to another website for processing shipping and payment. If we need to check the status of shipping and payment, we may go to the website of a transportation company or a bank instead of directly go back to the original E-commerce website. In the viewpoint of Service Request, the service like this is totally a mess because people will buy not only one product from Internet E-commerce website. How can we suppose that people can remember which shipping website or bank website he/she should go?

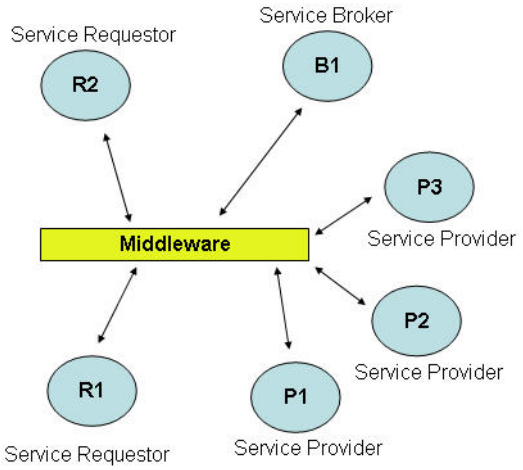

Figure 2. Service-Oriented Middleware

We have met the similar situation in our software development environment. Shown in Figure 2, Service Requestor, R1, connects to Service Providers in the sequence of P1 and P2. According to basic SOA model, R1 has to get the addresses of P1 and P2 from Service Broker, B1. Therefore, R1 can connect to the service in the sequence of P1 and P2. If one day the service makes some modifications because of organizational rules or policies, the sequence of the service is changed to P2 and P1. Then, R1 must change the programming codes to meet the new service requirement. Similarly, if the service needs a new Service Provider, P3, to join the service

sequence, then Service Requestor must face many programming re-coding issues.

In our service-oriented middleware, we fix the problems discussed above. The middleware holds the following characteristics:

1. Configurable sequence of a service [9].
2. Service Transparency to client applications.
3. Auto-discovery/ Auto-changeable mechanism of Service Providers

To achieve above characteristics, we design service metadata that is composed by XML syntax and service pipeline, which are discussed in the following sections.

3. Service Metadata

In this section, we discuss about what the service metadata is and how it is used. Service metadata is a collection of data written in XML. The purpose of metadata is to help middleware decide and make a correct service sequence which is composed by Service Providers. Although Service Requestors can retrieve connectivity information from Service Brokers, Service Requestors do not have any information to make a correct service sequence for a service. For example, a Service Requestor can retrieve the services of shipping company and payment component. Generally speaking, service requestor will connect to Payment Company first to ensure that the credit card or check validation of customer is passed successfully, and then it connects to shipping company to ship the goods to the customer. Due to the needs of different business processes, the shipping operation may be run before payment operation based on the trust relationship among companies. This sequence – payment first and shipping second – may be inversed. Therefore, client application developers have to re-organize their programming codes to meet the new requirement of business processes.

In order to ease the complex work for client applications, we introduce the service-oriented middleware to do the work of composing correct service sequence. Every service provider will generate its own service metadata and send the metadata to Service Broker. The format of the service metadata is like as follows,

```
<Services>
        <Name></Name>
        <Description></Description>
        <ProvidingServices>
                <Service>
                        <Name></Name>
                        <Description></Description>
                        <Interface></Interface>
                        <ControlSequence></ControlSequence>
                </Service>
                <Service>
                        <Name></Name>
                        <Description></Description>
                        <Interface></Interface>
                        <ControlSequence></ControlSequence>
                </Service>
                <!-- A collections of Service -->
        </ProvideingServices>
</Services>
```

Figure 3. Service Metadata

One Service Provider may provider more than one service. For example, a railway company may provide not only the service of passenger transportation but also the service of freight transportation in the same railway system. Therefore, service metadata carry the information of services that are provided by that organization. In the segment of 'ProvidingServices' in service metadata, there are 'Service' collections with other data – service name, service description, service interface, and service controlsequence. Here, we will emphasize the significance of the information of service interface and controlsequence. Service interface is the endpoint address to indicate that where other applications connect to this service. By means of XML web services, service interface is designed to appear as a web address. For example, http://seweb.csie.ndhu.edu.tw/tuitionservice1.wsdl. Service controlsequence is the right of priority while middleware decides which service has the best quality.

```
<Services>
        <Name>Taiwan Railway Company</Name>
        <Description>Taiwan Railway Company provides two services, passenger
                transportation and freight transportation.</Description>
        <ProvidingServices>
                <Service>
                        <Name>Passenger Service</Name>
                        <Description>Passenger Service will help you to collect
                                the train information you want. You can
                                order the ticket and check the availability
                                status of the train.</Description>
                        <Interface>http://www.trc.com.tw/passenger.wsdl</Interface>
                        <ControlSequence>5</ControlSequence>
                </Service>
                <Service>
                        <Name>Freight Service</Name>
                        <Description>Freight Service will help you to collect
                                the tranin information available to freight
                                transportation. You can order or check the
                                available spaces of the train.</Description>
                        <Interface>http://www.trc.com.tw/freight.wsdl</Interface>
                        <ControlSequence>6</ControlSequence>
                </Service>
        </ProvideingServices>
</Services>
```

Figure 4. Sample of Service Metadata

The service metadata is generated by Service Provider and sent to Service Broker. Service metadata then is saved in the repository. Now, service-oriented middleware will connect to Service Broker and check the repository, and the middleware will choose the appropriate service according to the needs from client applications.

4. Service Pipeline

In this section, we introduce a service pipeline to organize the sequence of service. Service pipeline is a simple data structure that includes the sequence number and the interface of the service.

This service pipeline is maintained by the service-oriented middleware. Initially, the service sequence is constructed by middleware administrators according to the needs from client users. Middleware will search the repository in Service Broker for appropriate service based on keywords or descriptions. The data of interface and controlsequence of service metadata will be selected and sent to middleware to be the part of service pipeline in the middleware.

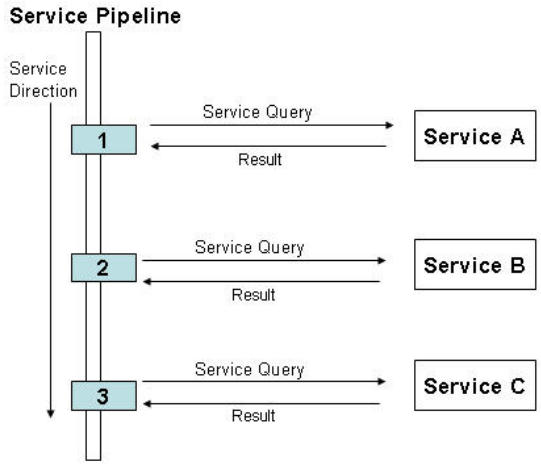

Figure 5. Service Pipeline

The middleware may select several similar services. According to the data in service metadata, a service will be used by the middleware based on the value of controlsequence property. The larger number it is, the larger probability it is adhered to the service pipeline. Also, the service-oriented middleware introduces an agent for the work of composing the sequence of the services. In our implementation, the agent is an independent service program working with the middleware. The agent is responsible to several jobs, e.g. composing the sequence of services according to the needs from client users, automatically transferring data among the services for data consistency, and data backup and restoring work. Therefore, when the agent needs to re-compose the new sequence of the services, the only thing it needs to do is change the interface value for the steps in the service pipeline. For example, as shown in Figure 5, step 1 in the service pipeline is to connect to Service A by the interface value that is recorded. After step 1 is finished, step 2 is ready to connect for next operations. If we need to exchange the sequence of the services, the agent just exchanges the interface values for step 1 and step 2.

Based on the service metadata, service pipeline, the agent of the service-oriented middleware, we can achieve the three characteristics that is mentioned above of the service-oriented middleware.

5. Middleware Development

The service-oriented middleware is developed by Microsoft .Net Framework [4] and C# language. By functionality, we design three areas of the middleware – web services area, management area, and core area as shown in Figure 6.

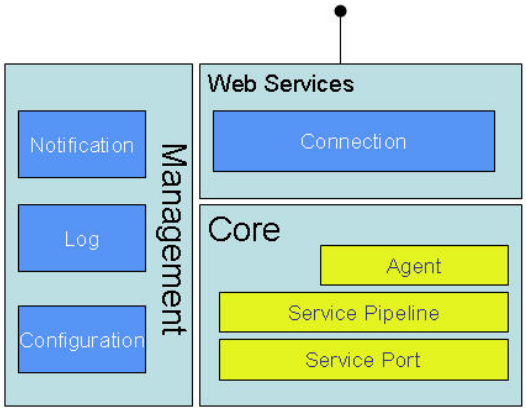

Figure 6. Middleware Architecture

The web services area in the service-oriented middleware to responsible to communicate with client applications. It provides one entry point to the middleware. The connection component is the only one component in this area and it is responsible to user authentication. If a user is authenticated, the request will be forward to Agent [8] in Core area.

In management area, there are three components – Notification, Log, and Configuration. Notification is responsible to notify administrators or maintainers when the service-oriented middleware meets any exceptions or errors. Now, we implement e-mail functionality to notify administrators or maintainers. It is also possible to develop other way of notification in the future, such as mobile phones or instant messenger applications. Log component is responsible to log any event that we want to record in the middleware. For example, when a user is authenticated, this event will be logged by Log component. The log messages are written to a log file for operational and security consideration. Configuration component is responsible to get and save system-level settings, for example, e-mail server

address, backup/restore path and device. The configuration component will save the system-level setting in a configuration file in XML syntax.

In core area, it contains the agent, service pipeline, and service port components. The agent and service pipeline are mentioned in previous section. The agent is implemented as a Windows Service program which runs as a standalone process in Windows Server operating system. Therefore, the agent is always monitoring the data passing between client applications and the middleware and between the middleware and the Service Providers or the Service Brokers. Service port component is responsible to communicate with the services that are provided by the Service Providers and communicate with the Service Brokers to search the information of services. Therefore, service port component must understand all communicating data formats which is constructed by the Service Providers. For example, in one of our scenarios, the middleware must transfer employee data in the human resource system to the resource pool in the project management system. The service port component retrieves the data from human resource system and re-organizes the data in another format that is known in project management system. This is the example of purpose of service port component for data consistency. Of course, the data transfer activity is triggered by the agent program. The agent program is the heart of the service-oriented middleware.

5. Conclusion and future work

The middleware contains not only the traditional functionality but also service-oriented architecture. By using this middleware, client applications can easily get the services which are a collection of services provided by the Service Providers without knowing too much detail about the interoperability issues among the Service Providers. In addition to this, we also combine the idea of the agent to this middleware and make the middleware become smart and automatic. Now, this service-oriented middleware is running and testing in a software start-up in Taipei, Taiwan for internal and research use and it may be packages as a commercial product in the future.

In our service-oriented middleware, we build service metadata and service pipeline for the Service Providers and the middleware respectively. Also, the Service Brokers need to save the service metadata in its repository. In the market, UDDI is one of the choices for the Service Broker. In the mean time, the service metadata repository and searching mechanism are not integrated with any UDDI server products. Our future work may integrate the service metadata into one UDDI server and make this middleware involve in other technology, for example, Pervasive Computing [7,10].

7. References

[1] Nirmal K Mukhi, Ravi Konuru, Francisco Curbera, “Cooperative Middleware Specialization for Service Oriented Architectures”, 13th International World Wide Web Conference, New York, USA, May 2004

[2]XML. Published by W3C at http://www.w3.org/XML

[3] Java Application Server. Published by Sun Microsystems Inc. at http://java.sun.com/

[4] Microsoft .Net Framework. Publish by Microsoft Corporation at http://msdn.microsoft.com/netframework/

[5] Open DataBase Connectivity. Publish by SQL Access group in 1992.

[6] Web Services Description Language. Published by W3C at http://www.w3.org/TR/wsdl

[7] Tatsuo Nakajima, “Middleware Design and Human Factor”, Ninth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2003.

[8] Noh-sam Park, Gil-haeng Lee, “Agent-based Web Services Middleware”, IEEE International Communication Conference 2003.

[9] Nikola Milanovic and Miroslaw Malek, “Current Solutions for Web Service Composition”, IEEE Internet Computing Magazine, Nov-Dec 2004.

[10] Jon Robinson, Ian Wakeman and Tim Owen, “Scooby: Middleware for Service Composition in Pervasive Computing”, 2nd Workshop on Middleware for pervasive and ad-hoc computing, 2004.