Research on Construction of EAI-Oriented Web Service Architecture[1]

Xin Peng, Wenyun Zhao, and En Ye

Software Engineering Lab, Fudan University, Shanghai, 200433, China
cspengxin@163.com, wyzhao@fudan.edu.cn, yeen@vip.sina.com

Abstract. Web service based data exchange has been the trend of EAI. We can create Proxies to expose existing systems as web services. There have been some tools that can help users to construct web services on the basis of existing systems. However, authority control and dynamic service configuration are not taken into account in the process of construction. In this paper we present an EAI-Oriented unified authentication model, and then put forward the service-items-based authority control and dynamic service configuration management in the construction of EAI-oriented web service architecture. Tool support in the process of construction is also discussed.

1 Introduction

One trend of EAI is the data exchange service based on web service. It supports the data interaction not only between systems in the same kinds of platforms but also between the cross-platform systems. A usual method of adopting web service is to add a layer of web service proxy on the basis of the original systems and then convert the systems into web services [1].

XML-based web service can be seen as a representation layer which is from program to program [2]. The business service provided as web service is aimed at external applications and hence should be able to deal with the various and changing service requests. At the same time, web service is also required to construct the authority system of the service requesters, which can guarantee the security of data exchange.

In the environment of EAI, each service publisher can integrate and publish local services on the basis of local business services and external services. The integrated elements are much more complicated. Therefore, it is important to make interdependent web services in the EAI system evolve controllably and keep dependable.

We introduce service configuration management in EAI-Oriented Web Service Architecture. All the local business components and external web services needed will be put under configuration management. Each service item is published through the unified service interface, which acts as a proxy to expose inner services.

[1] Supported by the National High Technology Research and Development Program of China (863 Program) under Grant Nos. 2002AA114010; 2001AA113070. It is also Supported by Shanghai Technology Development Foundation (No.025115014).

M. Li et al. (Eds.): GCC 2003, LNCS 3032, pp. 592–600, 2004.

Some recent tools can help to perform the construction of web service based on existing systems automatically. For instance, the WebSphere platform can help users to create service proxies for CICS middleware, Java class and EJB automatically [1]. The paper [2] designs a tool to help publish web service by constructing service packages acting as different roles. The constructions of web service discussed above do not deal with the needs of authority control and dynamic service configuration.

This paper is organized as follows. Section 2 presents a unified authentication mechanism in EAI. Section 3 describes the unified data service interface. Section 4 discusses an EAI-oriented web service architecture of authority control and service publication based on the diverse granularity configuration items.

2 EAI-Oriented Web Service Unified Authentication

It is thought in [3] that the web service model supports not only the reuse of software but also the reuse of the operating environment. And it proposes that authentication should be abstracted as a component and act as a web service of the third party.

The systems operating independently have their own system of authentication. Therefore, the most important task for EAI is to build a unified authentication system in the enterprise. The unified authentication system must provide unified user authentication and role management at the enterprise level.

We bring forward an EAI-oriented unified authentication model of web service (Figure 1). The authentication server deals with the requests of authentication and the role management unifiedly.

In the unified authentication system the role is managed unifiedly. Each service provider has its own authority control system, which determines the corresponding authority of different roles in local service. The security of the whole transfer channel and data transfer itself can be solved by adopting techniques such as SSL and PKI, which are discussed in [3],[4].

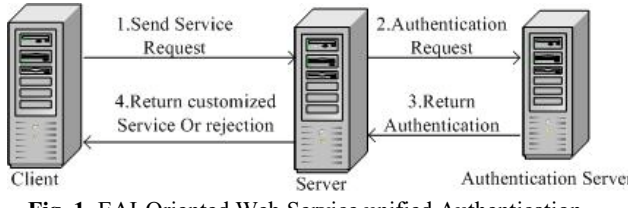

Fig. 1. EAI-Oriented Web Service unified Authentication

3 Unified Data Service Interface

The paper [2] thinks that what the web service should do is to provide information to the invoker and transfer the received information to the underlying system. When a web service is introduced into a system, the web service should simply form another layer, and play a role analogous to and parallel with the user interface layer [2]. There-

fore, a proxy layer should be deployed on the basis of the original application, which is responsible for the sending and receiving of data. Meanwhile, authority control and service customization should be carried out.

The paper [2] designs two data service interfaces (.Net interface), which aim at a single method invocation and method sequence invocation separately (Figure 2). In the interfaces, the Get and Post method are used to receive and set the data. The unified service interface serves as the web service proxy, which receives the service request of the client, drives the local services to execute according to the sequence and sends the return data as a unified interface.

```
public interface IDataService
{
  object GetData(string method,object[] values);
  bool PostData(string method,object[] values);
}

public interface ISequenceDataService
{
  object[] GetSequencedData(string[] methods,object[][] values);
  bool[] PostSequencedData(string[] methods,object[][] values);
}
```

Fig. 2. Data Service Interfaces

The advantage of this design is to improve the abstraction level of service interface. Moreover, the method sequence reduces the amount of messages between the client and the server and thus decreases the occupation of bandwidth [2].

4 Management of Diverse Granularity Service Items

4.1 Web Service Items Management and Configuration Management

In the web service based on data exchange, the core is the data items that service requestors concern. In our web service system, service items are those provided public methods. And a service invocation of the client is composed of a sequence of invocations of public methods.

In the configuration management, the basic unit is configuration item. And software configuration is the composition of configuration items. Configuration is the significant composition of configuration items. After the development, products can be managed by version-publishing, which is the composition of baselines.

4.2 Authority Control Based on Public Service Items and Their Interfaces

In our web service system, the service interface published publicly is the smallest unit invoked by the client. It is also the basic unit of authority control. On the basis of it, the method parameters should be controlled further. It is because that the selection of parameters determines the data returned or the inner executing mechanism.

We should first set up the authority distribution records. The authority control divides into two levels: method level and parameter one, which should be both maintained by every service provider. The control at the method level only needs to check whether the authority list of the method includes the role of the client. Contrarily, the control at the parameter level is complicated because the format of parameters is different and the authentication principles are various.

As illustrated in Figure 3, the process of authentication is that the component of authority control returns the results to the unified interface component after the authority authentication at the method level and parameter level. Only the information of roles accepted by every method is required to be registered in the authority list at the method level. At the parameter level, however, the authority information in every method should be registered. These methods take the parameters required when the client invoke them as their own parameters and return the result of authentication.

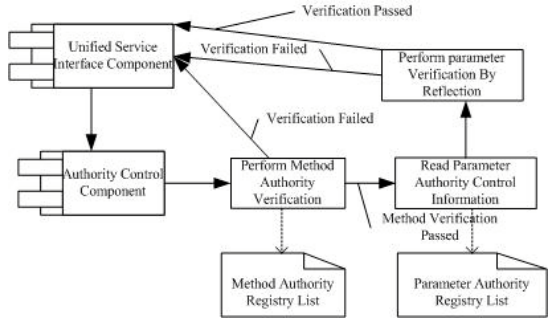

Fig. 3. The Process of Authority Verification

We define two java interfaces for authority verification (Figure 4). The class performing authority verification should implement the interface IServiceCheck, which defines three methods. The checkInvokeMthod method and the checkInvokeParams method perform authority verification at the method level and the parameter level separately. And the checkInvokeSequence method verifies sequencing of methods. The sequencing verification will return false as long as any method verification in the sequence fails. Considering that a user can own several roles, the parameter role is an array.

```
public interface IServiceCheck
{
 public static boolean checkInvokeSequence(String[] methods,object[][] values,String[] role);
 public static boolean checkInvokeMethod(String method,String[] role);
 public static boolean checkInvokeParmas(String method,String[] role,Object[] values);
}

public interface IMethodParamsCheck
{
 public static boolean checkInvoke(String role,Object[] values);
}
```

Fig. 4. Authority verification interfaces

IMethodParamsCheck is an interface that each class defining the rules of verification at the parameter level should implement. In a concrete implementation each method to be published should define a corresponding class implementing the IMethodParmasCheck interface to perform parameter authority verification and register the class in the service publishing information.

4.3 Web Services Publishing of Diverse Granularity

WINGCM [5][6] proposed the concept of software configuration management of Diverse Granularity. The issue of granularity also exists in web services. Web services of coarse granularity can make the services more efficient and the corresponding invocation will be simple. However, web services of fine granularity can present more flexibility, since the client can assemble the service sequence flexibly.

We create proxies of various levels to provide web services of diverse granularity. Figure 5 presents the architecture of diverse-granularity web services publishing system. The Service objects represent web service items exposing existing application to the clients. They are proxies of the business services and invoked through the unified service interface by reflection mechanism. The Class objects are class components encapsulating business logic. In the situation described in figure 5 Class 1 represents business logic of high abstract level and is published through Service C. At the same time Class 2, 3 and the EJB components which compose the bottom service of Class 1 can be exposed independently through Service A, B and D to provide services of finer granularity. These business components can also invoke web services provided by other server to compose local service.

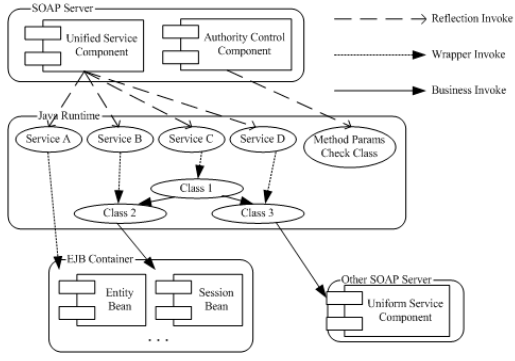

Fig. 5. Diverse-granularity web services publishing

4.4 Dynamic Service Configuration Management

Each service item invoked through the unified service interface is composed of correlative bottom components directly or indirectly. A concrete invoking process concerns certain versions of bottom components. So the evolvement of any component

involved in the entire service system will lead to the evolvement of versions of some services. In our web service system the configuration item and the service item are defined as follow:

Configuration item: java classes, EJBs, or components of other types at various abstract levels, which compose the entire service system. The web services provided by other servers can also become configuration items.

Service item: The public methods exposed through the unified service interface. They will be invoked by reflection mechanism.

A configuration item is the basic unit of service configuration management. Atomic configuration items can be an EJB, or a java class or a class package which acts as a single component entirely. Especially, web services invoked from other servers will be treated as local atomic configuration items. A configuration item is also a unit of maintenance and version evolvement, which means an atomic configuration item should be modified or replaced as a whole. Each configuration item has a unique and unalterable identity including external services. Configuration items are stored in the repository according to the identities and can be gotten by certain way. External services are special configuration items since they can only be maintained on other servers. Information about the external services will be recorded locally. The evolvements of external services can only be informed by external entities, such as the private UDDI registry.

A service item is a public method that acts as the proxy of certain bottom components. It is the basic unit of authority control. The wrapping components providing these methods are configuration items of the highest abstract level. Correlative information about service items of each web server in the EAI system, including the description of service classes, methods and the parameters, should be published to the private UDDI registry of the enterprise.

A service configuration management layer should be added into our web service system (Figure 6). The aim of that management layer is to guarantee that the service items under dynamic configuration evolve correctly and keep traceable. The repository in software configuration management maintains resources in the project development, while the repository in our service configuration management takes runtime components as resources. Various versions of the components are stored in the repository according to the version number. The version of certain component is determined by not only the component entity itself but also the versions of bottom components concerned. So the evolvement of a component will result in the evolvement of components at higher abstract level, which depend on the component directly or indirectly.

The objects that web service configuration management aims at are runtime components, such as java classes or EJBs. There exist two kinds of configuration items. One of them is service components developed and maintained locally. The management of these components should integrate SCM tool, which manages the development resources such as document, source code etc entirely. While the service publishing will be managed by the service configuration tool. In addition to delivery of source code the “check in” operation of components will contain compiling of the code to get runtime components. Sometimes deployment will be involved too. There still exists another kind of components. They are entirely runtime components. The

service administrator can perform management directly by the service configuration management tool.

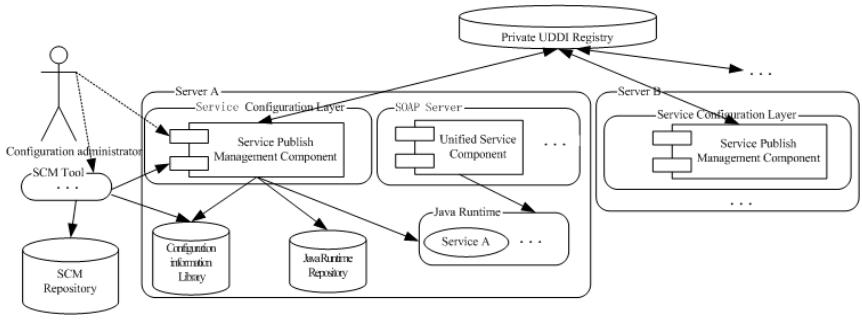

Fig. 6. Web Service Configuration Management Model

The relations between configuration items also need systemic management, most of them are dependencies of various kinds. The version evolvement of certain component will make the components invoking its services directly or indirectly evolve in turn. The whole historical information in the dynamic configuration process is stored in the configuration information library (Figure 6). A service item of certain version is composed of components of certain versions from the bottom up. Therefore according to the history of version evolvement, the dynamic configuration information of service items can be traced and service items can smoothly evolve or roll back.

Consistency control of the evolvement of configuration items should be performed. For example, the tool can use reflection to check the downward compatibility of new versions of components. This can guarantee that the interfaces referred by other components are still available in the new version. On the other hand the service items, which are configuration items of the highest abstract level, determine the versions of all the components involved directly or indirectly in the service. So it is possible that version conflicts exist between several service items. For example, two service items may have different version demands for the same component, then this is a configuration containing conflicts. The solution can be that a service item adopts another version or imports new component to replace the conflicting component. It should be illuminated that the version of a service item is determined directly by the wrapping class which the method belongs to. It is the wrapping component that is the object of configuration management.

4.5 Dynamic Service Configuration and UDDI Registry

Web services provided by other servers are difficult to be involved in the service management, because the versions of external services can hardly be managed under local control. The enterprise private UDDI registry can then act as a coordinator of the servers. When publishing services each server should register information about the services on the UDDI registry, such as involved classes, methods, parameters and the versions. The function of the UDDI registry should be extended to provide additional coordinating service. When a server needs to invoke services of others, the configura-

tion administrator should register this inter-server service requirement on the UDDI registry by using service configuration management tool. In this way when a server registers a service version evolvement the registry can inform those servers invoking this changed service. Thus the scope of service configuration management is extended and the entire EAI system can highly cooperate with each other.

5 Summary and Conclusions

In this paper, we introduce some concepts in SCM, and present the service item based authority control and the dynamic service configuration management architecture. In such EAI system each integration unit represents more coordination and EAI is implemented by services exchanges between various servers. The unified authentication is performed on authentication server, and each web server constructs its own integrated authority control system under the unified role definition. Then the authority control can be performed at the method level and parameter level. At the same time, the publishing of web services will be maintained under systemic configuration management. Certain versions of components involved in exposed services compose the various service items of corresponding versions, which can be invoked through the unified service interface.

Such web service architecture makes a high request for the interaction of integration units and meets the needs of the EAI within enterprises better.

The construction of the entire web service system needs the support of tools, including the SCM tool, the web services configuration management tool and the service construction tool. [2] presents a web exposer tool which can generate web services on existing business objects. In our future work we will research on these support tools. One of our research teams has developed a diverse granularity SCM tool [6]. We will extend support interfaces on the basis of the work and make the SCM tool integrate seamlessly with the future web service configuration management tool. Integration with the enterprise component library is also a significant orientation. If enterprise component library, SCM tool and web service management can be integrated seamlessly, it is out of question that the resource management of the enterprise can be promoted greatly and the software reuse can exert greater effect.

References

1. Litoiu, M.. Migrating to Web services - latency and scalability, Web Site Evolution, 2002.Proceedings. Fourth International Workshop on , 2 Oct. 2002
2. Nicoloudis, N., Mingins, C.. XML Web services automation: a software engineering a proach, Software Engineering Conference, 2002. Ninth Asia-Pacific , 4-6 Dec. 2002
3. Baldwin, A., Shiu, S., Mont, M.C.. Trust services: a framework for service-based solutions,Computer Software and Applications Conference, 2002. Proceedings. 26th Annual International , 26-29 Aug. 2002

4. Song Zhiqiang, Chen Huaichu, Shen Xichen. An Uniform Authentication Architecture in Campus Network and an Implementation of Application Roaming Based on the Architecture, COMPUTER ENGINEERING AND APPLICATIONS, 2002Vol.38No.20
5. Software Engineering Lab, Fudan University. WINGCM technology report
6. Ke Liping, Zhao Wenyun, Zhang Zhi. R&D on Process-Based Software Configuration Management, COMPUTER ENGINEERING, 2003 Vol.29 No.5