

MDESB: Model-driven Enterprise Service Bus

ZHANG Ming-bao

Department of Information Management and Electron Business,
Nanjing University of Aeronautics and Astronautics, Nanjing, China
zmb7381@126.com

Abstract—Traditional Enterprise Service Bus (ESB) does not consider the impact of objects of different granularity on Enterprise Application Integration (EAI), lacks service description means of semantic level, and only uses manual methods to implement the coordination of message format conflict, which results in the traditional ESB application in EAI very difficult. In order to solve these problems, the idea of Model-driven Enterprise Service Bus (MDESB) is put forward, and based on this idea this thesis sets up an architecture of MDESB and introduces its working mechanism.

Index Terms—MDA, SOA, ESB, EAI

I. INTRODUCTION

The idea of EAI is to optimize enterprise information systems and information resources utilization [1]. Early EAI[2] uses point-to-point model to implement system integration, but because of its low efficiency it is replaced soon by point-to-face integration model. The point-to-face integration model emphasizes utilizing component technology and open interface standard to realize enterprise's integration aims, and some classical IT integration solutions, include CORBA-based, J2EE-based and .NET-based solution, gradually appeared in this stage[3,4]. Although now these classical solutions have been applied very widely, they bring some new integration problems. For example, how to integrate CORBA-based system to .NET-based system easily, how to cross firewall when use RMI or IIOP in information system communication. In order to remedy these problems and following along with the development of SOA architecture, one kind of new EAI solution named as ESB is raised, which mainly utilizes SOA technologies to solve EAI problems.

ESB is software infrastructure that simplifies the integration and flexible reuse of business components within a service-oriented architecture. An ESB provides a dependable and scalable infrastructure that connects disparate applications and IT resources, mediates their incompatibilities, orchestrates their interactions, and makes them broadly available as services for additional uses. The basic functions of ESB include message transformation, service call, application integration, service quality, security service and system management[5-7].

Now regarding ESB system architecture and functions have basically reached a consensus in theoretical circle, but using ESB to realize actual EAI aims is still a difficult problem and yet to be adequately resolved. There are several works to get around this problem. Article 8 indicates that because ESB

transforms all different levels of interactive interface only to web service that makes ESB based EAI face severe challenges on the lack of simplicity and reusability and so ESB needs to consider system integration problem simultaneously at three different levels that are business process layer, transformation layer and transmission layer. Paper 9 proposes an idea of data service bus which discusses the importance of integrated data layers and describes an ESB based data integration method. Paper 10 analyzes the drawbacks of traditional ESB using XSL to realize manual message format conversion and points out the requirements of semantic Integration in EAI.

Based on above researches, this paper proposes an idea of Model-driven Enterprise Service Bus (MDESB) which uses the model-driven software implementation method to design ESB. Section II summarizes the challenges faced by ESB applied in EAI scene, Section III introduces the idea of MDESB, Section IV describes the system architecture of MDESB, and Section V provides a summary and a discussion of some extensions of the paper.

II. CHALLENGES OF ESB APPLICATION IN EAI

In essence, current ESB is only an operational level EAI infrastructure. Compared to the EAI platform built on middleware technology, The biggest feature of ESB is that its use of SOA related technical standards makes ESB receive a much broader cross-platform features, as well as a more loosely system coupled characteristics. After ESB's success in solving integration problems at the technical level, how to optimize EAI design and enhance the efficiency of EAI design will become a new problem to be solved urgently. Now using ESB to build EAI system still rely on engineers' deep understanding of integrated objects as well as their careful design, which has led to ESB application design more difficult. In addition, as the ESB applications become more and more in-depth, EAI systems become more and more complex, which in turn further increase the difficulty of EAI system design. In order to expand abilities of ESB to simplify EAI system design, the following problems must to be considered.

A. Integrated Object Granularity of EAI

With the types of enterprise application system become more and more diversified, as well as a variety of advanced enterprise computing technologies become more and more common, ESB-based EAI need to consider a wide range of integrated object. Figure 1 shows that enterprise application system consists of four different layers. Data layer refers to a variety of data stored in the

database; Component layer includes function modules, components or web services, etc. which implement special logic process to accomplish a particular function and need to use the objects of data layer. Application layer refers to a variety of enterprise applications which realize their own functional by their call to functional modules or components of component layer. Process layer figures to all kinds of business processes which operated by process management system and whose function are achieved by the integrated use of application system of application layer.

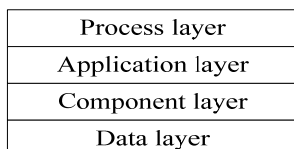


Figure 1. four layers of enterprise application system

The objects of each level in figure 1 all should be integrated to ESB, which has two main reasons. First of all, the "open" characteristics of modern enterprise computing technology means that the object of each level all can provide an open, standard external interface. For example, at the data layer, SQL, ODBC, JDBC and other technologies can be used in a database call; At the component layer, CCM, EJB, DCOM, Web service and other technologies can be used to implement the call of function modules; At the application layer, a wide range of open programming interface provided by applications system can be used by other systems to communicate with themselves, such as the use of Business Application Programming Interfaces (BAPI) implementation of the SAP; At the process layer, the workflow management technology can be used to package, describe and control process. In short, the existence of each object-level interface provides a technical guarantee for application integration. Secondly, because of the complexity of enterprise applications system, there are the objective needs of integration for the objects of each level. For example, a sporadic demand of enterprise applications may only need to query some database data, while other demand may need to use some functions of components units, there is also possible sometimes that a certain demand requires the integrated use of the different objects of all four levels to complete specific administrative tasks.

ESB only provides a support means of application integration at the component layer, which implements interactive process of component and management of these process through mapping the external interface of components to a web service, and for data integration, application integration and process integration, the integration process can be come true still through interactive web service implementation. Due to such characteristics of ESB, designers need to carefully design such a transformation process and require additional design-aided management system when using ESB in EAI. For example, data integration aims can be achieved by using web service to package database access interface, but it only implemented a data access operation,

the integrity of the database access control, security control and so on problems still need to design additional control system to solve; for application integration, a cross-system call also can be realized by using web service to package application system external interfaces, but how to process the relationship between application system and web service interfaces as well as the relationship between different web services still need amounts of difficult and meticulous work of designer. Moreover, some important functions involved in process integration such as process description, process implementation control, transaction processing and so on are still not supported by ESB.

B. Service Description and Message

One important problem to be solved by EAI is implementation of interactions between different information systems, and now ESB uses interaction of web service to solve such problem. Web service implements the standardization of call interface, but this implementation is limited only to grammar level and is lack of standardization of the semantic level, which makes the use of ESB face two problems.

First, while the ESB provides a UDDI-based service registration and service discovery mechanism, but because the service descriptions based on WSDL only contains operation-related information, such as service name, calling parameters, the key quality indicators and other items, and does not contain application-related information, such as service functions, application context and so on, so it is very difficult for designer to understand service functions and it's application methods. In addition, the UDDI-based service discovery mechanism only uses a "word-shaped matching" search method, which makes the ability of service discovery is weak. The above-mentioned problems still need system engineers to deeply understand the internal working mechanism of integrated object, which causes the ability of using UDDI to build "loosely coupled" system not to be given full play.

Second, ESB using message mechanism to implement the call of Web service brings some problems still not to be resolved. Message consists of two aspects, namely, grammar form and message semantic content. The current ESB uses XML to describe message exchanged between services which only implements the standardization of message grammar formats. Moreover, ESB uses XSL technology to realize the mapping of contents and formats between different messages, and this mapping set up through "one-on-one" conversion model which mainly rely on designers' understanding of related messages. Such "one-on-one" conversion model is very inefficiency and lack of reusability. For the field of B2B integration, the impact of this problem even more serious because of the border cross-enterprise collaboration application. At present, many international organizations are actively setting up a variety of information exchange ontology or metadata used in different fields, such as ebXML, RosettaNet, BizTalk, OAGi and so on. One possible method is that these existing information exchange ontology may be used to

semantic coordination of exchanged messages based on "one-to-many" model, but at present there is little concern about such questions in ESB Research.

C. EAI-based Application Integration

ESB application will reduce the difficulty of EAI, which will make enterprise-level EAI applications gradually increased. In many cases, the existing EAI application will be applied as objects to be integrated by new EAI plan, which causes a new problem that is EAI-based application integration. Because the running of EAI application involves many different data objects, components, applications and processes, as well as lacks of a clear boundary, so it is very difficult to be integrated by other EAI applications. EAI-based application integration needs to consider some factors as follows.

First of all, description of the integration object. EAI-based application integration need to deal with five types of integration objects, that is, data, components, applications, processes and EAI applications. The five categories of object have to provides standard open interfaces, which should be oriented ESB, to facilitate the external call. In order to achieve effective reuse, it is necessary to standardize the description of these five categories of object, and these description content may include each object's functions, operating parameters, the context of running environment, Pre-operational preparation rules, output treatment methods, etc.

Second, description of relationships between integration objects. An EAI application can be seen as a collaborative process of such five types of object. While there are complex call relationships between the five categories of object, for example, component object may call some data objects, application system may access some components objects and data objects, business process object, which is essentially workflow scheduling, may use some application systems, components and data objects, EAI application object may be related to the other four objects. Therefore it is very important to describe these relationships for building an EAI-based application integration. Such relationship description has two benefits, one is that can avoid the emergence of various conflicts, the other is that integration can achieve the optimal path.

Third, description of EAI integration mechanism. Because of very complicated integration process involved the five categories of object, ESB environment should provide simulation / testing tools to verify the effectiveness or find problems of EAI-based application integration. Implementation such functions requires to achieve a clear description of EAI application mechanism and still to set up it's modeling methods.

III. THE IDEA OF MDESBS

In order to solve above problems, reference OMG proposed MDA method, a new idea of Model-driven Enterprise Service Bus (named as MDESBS) is suggested. MDESBS considers a variety of EAI resource models as it's core, and emphasizes the full use of basic function of existing ESB infrastructure. MDESBS achieves its

functional goal mainly through the expansion of modeling environment and model control mechanisms based on existing ESB functions. MDESBS have the following characteristics: a) the process of EAI is described by different of models that means user can achieve effective management of the entire life cycle of the EAI process by managing model; b) the model is adopted to control the operation of the EAI process which can simplify user's works of simulating and monitoring EAI running ; c) model reconstruction is adopted to efficiently implement the EAI application re-engineering and the reuse of EAI resources; d) standard resource modeling techniques is used to implement automated and semi-automatic conversion between different models at all levels ,as well as between models and their implementations, which can enhance capabilities of system modeling and development.

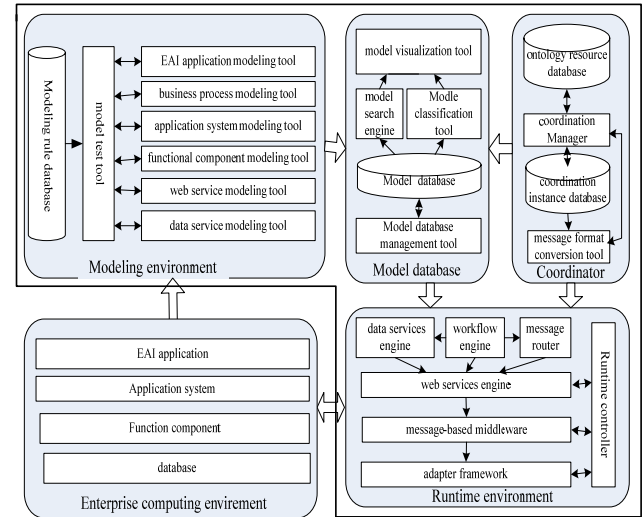


Figure 2. Architecture of model driven ESB

IV. system framework of MDESBS

The system architecture of MDESBS is shown in Figure 2. It includes four components which are modeling environment, model database, coordinator and runtime environment.

A. Modeling Environment

Modeling environment provides tools to describe a variety of EAI resources by using corresponding modeling language. A variety of resources are abstracted into six types of modeling objects: data service, web service, functional component, application system, business process and EAI application, and modeling environment provides corresponding six types of modeling tools. In these modeling objects, data services and web services are two essential component and other modeling objects are constituted by the composite use of these two basic components. These modeling objects are introduced in order to adapt to EAI modeling at different levels, at the same time in order to clearly describe the complex relationship between models at different levels. Taking into account the complexity of EAI modeling, EAI modeling environment still includes a model test tool which provides the following functions: integration path optimization test, integration resource conflicts test,

resource package optimization test, security test and database integrity test. To carry out model testing, there is inevitable need to use various rules of models test. Modeling rule database is mainly used to store and manage these rules.

B. Model Database

Model database is used to store a variety of EAI modeling description document which has two important benefits, one is that it provides a unified view of available resources in EAI for designer and the other is that these stored models can control the running of EAI application. In order to manage and utilize the model database efficiently, some tools are still needed. Model search engine and classification tool can help designer to organize and rapidly find the necessary integration objects. Model visualization tool graphically displays the relationships between integration resources which assist designers to obtain a comprehensive understanding of enterprise EAI status quo, the availability of resources and such resources' relationships. Model database management tool have two mission: First, implementation of traditional database management function to help users set up and manage model effectively; second , providing function of loading and running the model to enhance the flexibility of EAI.

C. Coordinator

The coordination of the conflict is implemented mainly through the establishment of mapping rules that depend on the nature of both sides of the conflict. Because traditional ESB only provide manual method to set up such mapping rules, which lacks of means to improve the efficiency of setting up mapping rules and can not realize reuse of existing rules, so it's very inefficient. In order to solve this problem, as shown in Figure 2, a tool named as coordinator is designed, whose role is to provide support means in order to help designers improve the efficiency of the design of mapping rules, as well as through centralized storage and management of various types of mapping rule to achieve the reuse of these mapping rules. the coordinator is composed of four parts. Ontology resource database is used to store all kinds of EAI modeling language ontology as well as various fields of information presentation ontology or metadata for information exchange; coordination instance database is designed for storing all kinds of mapping rule, the database schema may reflect the linkages between mapping rules which can facilitate the reuse of these rules. Coordination Manager is to manage Ontology resource database and coordination instance database, which provides tools to help users more effective use of resources stored in these databases. The main function of message format conversion tool is automatic generation the XSLT document required by message exchange, which uses Coordination Manager to obtain required mapping rules as it's input.

D. Runtime Environment

Runtime environment is mainly used to load and run a variety of EAI model. Runtime environment shown in

Figure 2 contains all the underlying infrastructure of traditional ESB, such as web services engine, message router, message-oriented middleware, adapter framework, operation supervisor, and so on. In addition, the runtime environment also introduces data services engine and workflow engine. data services engine is used to explain and implement description document of data services (i.e., data services model), workflow engine is used to explain and implement business process description documentation (i.e., business process model). Data services and business process is implemented by the use of web services to collaborate with required EAI resources.

V. CONCLUSIONS

In order to solve the new problems faced by ESB-based solution for the EAI, the idea of MDESB is put forward here. Our work shows that the MDESB implementation faces three main challenges. First of all, now technical standards for EAI modeling is not perfect, especially current modeling standards at the semantic level ,such as WSMO and WSML, are still in the process of being developed. Second, the data services technology is just emerging, data services architecture and technical standards are still at the stage of theoretical exploration; Third , even though there are existing lots of outcomes of open source which can be used for implementation MDESB, but integrated application of these results is still facing a lot of problems to be resolved.

REFERENCES

- [1] N Erasala, DC Yen, TM Rajkumar. Enterprise Application Integration in the electronic commerce world. Computer Standards & Interfaces, vol 25,pp:69-82,2003.
- [2] WA Ruh, WJ Brown, FX Maginnis. Enterprise Application Integration: A Tech Brief. John Wiley & Sons, Inc. New York, NY, USA. 2001
- [3] P Maheshwari . Enterprise application integration using a component-based architecture. Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International Volume, Issue, 3-6 Nov. 2003 Page(s): 557 – 562.
- [4] Vasconcelos, MM da Silva, A Fernandes. An information system architectural framework for enterprise application integration. System Sciences, vol 18,pp:267-283, Nov, 2004.
- [5] Anurag Goel. Enterprise Integration EAI vs. SOA vs. ESB. <http://hosteddocs.ittoolbox.com>. 2006.
- [6] Thomas Freund. ESB Interoperability Standards. <http://download.boulder.ibm.com/>,2008.
- [7] Dave Chappell. Enterprise Service Bus. O'Reilly. Jun, 2004.
- [8] Tariq Mahmoud, Jorge Marx Gómez. Integration of Semantic Web Services Principles in SOA to Solve EAI and ERP Scenarios; Towards Semantic Service Oriented Architecture. ICTTA 2008. 3rd International Conference on. 7-11 April 2008, On page(s): 1-6.
- [9] BEA. BEAAquaLogic Data Services Platform. <http://e-docs.bea.com/aldsp/docs25/index.html>,2008.
- [10] Peter Martinek1, Balazs Tothfalussy, Bela Szikoral. Semantically described services in the Enterprise Application Integration. 30th ISSE 2007:335-338.