UDC 004.732, DOI: 10.2298/CSIS1001031Z

Service-oriented Enterprise Interoperability in Automobile Supply Chain Management

Yong Zhang[1], Shijun Liu[1], Lei Wu[1], and Yuchang Jiao [2]

[1] School of Computer Science & Technology, Shandong University, Jinan, P.R.China, 250101
evelyn0330@163.com; {lsj,i_lily}@sdu.edu.cn
[2] SOA Design Center, China Software Develop Lab, IBM, Beijing, P.R.China, 100193
jiaoyuchang@gmail.com

Abstract. How to enhance interoperability between stakeholders and improve efficiency of supply chain management is the key issue that needs to be addressed in automobile industry. Existing interoperability solutions are suitable only for large enterprises, however, small and medium-sized enterprises (SMEs) lack cheap, easy to integrate and easy to customize solutions. This paper proposes a methodology that provides a guide on how to establish interoperability between enterprises through a federated approach. An interoperability service platform is designed and delivered in the form of Software as a Service (SaaS). This paper introduces the specifications of the service platform and proposes an interactive framework which is used to establish interoperability between the service platform and other on-premise applications.

Keywords: Enterprise Interoperability, Software as a Service, Supply Chain Management, SOA.

1. Introduction

Today an enterprise's competitiveness is to a large extent determined by its ability to seamlessly interoperate with others. Enterprise Interoperability (EI) has therefore become an important area of research to ensure the competitiveness and growth of enterprises [1].

In modern manufacturing field, such as automobile industry, an entire manufacturing process often cooperated by assembly factory and part suppliers. Agile supply chain management increasingly becomes an effective and important measure to enhance competitive advantage of enterprises that needs the support of agile information system to integrate their supply chain more effectively and quickly [2]. As auto manufacturers inexorably move their sourcing of components and low value-added operations offshore, to lower cost countries, so their supply chains increase in both distance and complexity. Many companies are faced with the challenge of providing an agile

response to customers and yet operating a lean operation across an extended global supply chain. This is a challenge that needs a solution beyond the abilities of simply judgment, the telephone and spreadsheets [3].

Enterprise interoperability issues in the automotive industry are important because the complexity of product, the design process, and the industry magnifies the impact of interoperability problems while obscuring their solutions. However, existing interoperability solutions are suitable only for large enterprises and SMEs lack cheap, easy to integrate and easy to customize solutions. Therefore, SMEs are still far behind in the reform of supply chain management due to their small IT budgets, crude process standards with little visibility data to enable them to share and compare with trading partners.

According to [13] and ISO 14258 [11], three categories of interoperability barriers are identified: conceptual barriers, technological barriers, and organizational barriers. What's more, there are three basic ways to relate systems together to establish interoperations: integrated approach, unified approach, and federated approach.

In this paper, our research work aims at developing a methodology that provides a guide on how to implement an interoperability solution in automobile supply chain management through a service-oriented and federated approach. More precisely the methodology allows establishing interoperability by: (a) constructing a virtual enterprise by identifying and involving various actors and stakeholders in an interoperability service platform; (b) dynamically configuring and composing available interoperability services according to identified requirements; (c) evaluating and improving the interoperability solution in practice.

To enhance interoperability among stakeholders and build agile supply chain for automobile industry, under the guiding of this methodology, an application platform named Supply Business Management (SBM) was designed and delivered in Software as a Service (SaaS) [5] business model, which is cheap, fast, reliable, and without major integration efforts, so they can be invoked by enterprises on the fly in support of their business activities.

In contrast to traditional on-premise software that is deployed and run in a data center at the customer's premise, SaaS software is run at a SaaS hosting provider and can be accessed via the Internet. SaaS offers many advantages for software customers. Instead of software licenses, maintenance and operational costs that occur in the traditional on-premise model, companies consume IT-services in the SaaS model can use the software on demand, just like they would use any other utility such as electricity or water. SaaS user pays only for the usage of the software for example in a pay-per-usage model.

The goal of SBM service platform is to provide a holistic solution enabling the collaborative supply chain management in a flexible and dynamic environment and especially to facilitate SMEs' participation to collaborative supply chain management processes. In SBM service platform, interoperability is considered to be a utility-like capability and delivered in the form of SaaS. This paper introduces the detailed architecture of the platform and proposes an interactive framework which is used to establish interoperability between SBM service and other applications.

2. Related Work

Since the beginning of 2000s, research on enterprise interoperability has been emerging. Most related work is concerned with the elaboration of an enterprise interoperability framework, such as LISI, IDEAS, AIF, EIF, etc. This paper followed the enterprise interoperability framework developed within the frame of INTEROP Network of Excellence (NoE). The purpose of this framework is to define the research domain of enterprise interoperability and help to identify and structure the knowledge of the domain. It has been considered that enterprises systems are not interoperable because there are barriers to interoperability.

In the NoE interoperability framework, there are many research challenges, interoperability service utility is just one of them. The delivery model of interoperability service is a key problem to be considered. On the other hand, Software as a Service (SaaS) is a new delivery model for software, which lowers the cost of development, customization, deployment and operation of a software application to support multiple tenants over the Internet. The SaaS vision focuses on separating the possession and ownership of software from its use. Delivering software's functionality as a set of distributed services that can be configured and bound at delivery time can overcome many current limitations constraining software use, deployment, and evolution.

A well designed SaaS application is generally distinguished by mainly three qualities: scalability, configurability, and multi-tenant efficiency [8] [9]. In [8] four maturity levels for SaaS applications are presented. [15] introduces an electronic contract management application which uses the third level of SaaS maturity model, so the scalability and deployment architecture are not mentioned. However, this paper introduces a SaaS application designed and delivered with the final level of maturity model; both loader balancer and fail over problems are solved.

Variation points have been introduced in [4] [6] [12] and it is one of the key concepts for software product lines to express variability. Variation points, often also called variability points or points of variability, allow the specification of points in a software design that can be customized in order to create new product line members. Similar to software product lines, SaaS applications have variability points where individual customers can customize the application to their needs.

In a SaaS application, the variability points exist in several layers, like UI layer, process layer, data layer, etc. [7] describes how these variability descriptors can be transformed into a WS-BPEL process model that can then be used to guide a customer through the customization of the SaaS application. [4] shows how the service component architecture (SCA) can be extended with variability descriptors and SaaS multi-tenancy patterns to package and deploy multitenant aware configurable composite SaaS applications. While in this paper, in order to realize the configurability, metadata is used to define all the variability points of the application. Meanwhile, JMX is used to manage the metadata so that changed metadata can be hot deployed immediately during runtime.

As far as integration is concerned, in [12] the need for integration of SaaS applications with existing backend systems is motivated and a framework is proposed. [10] uses executable EAI patterns as the solution for EAI problems thus leveraging on the work done in the EAI community for the integration of traditional applications. Our work differs from them as middleware, such as data synchronization toolkit and message engine are designed and developed to address the integration issues in business process layer and data layer.

3. Construction of Virtual Enterprises

The virtual enterprise (V.E.) concept is one of the most important ways to raise the agility and competitiveness of a manufacturing enterprise [13]. Under this concept a master company develops its products by using the manufacturing resources of external partners.

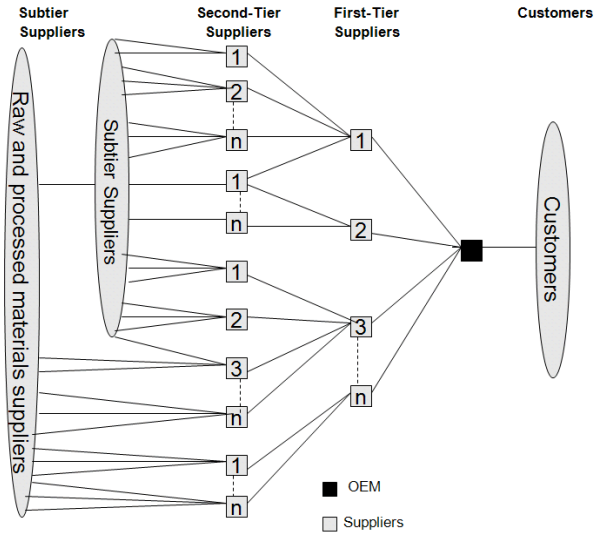

Fig. 1. Typical structure of the supply chain in automotive industry

In the process of designing and manufacturing an automobile, many individuals and organizations exchange product data. The design and manufacturing process involves many divisions within the original equipment manufacturer (OEM), many first-tier suppliers, a number of second-tier and sub-tier suppliers. This exchange of data supports the process of concurrent engineering and design, allowing these organizations to work together to improve the performance and manufacturability of a product and to advance the competitiveness of the industry.

Figure 1 shows a typical structure of the supply chain in automotive industry. As shown in Figure 1, the automotive supply chain consists of four

primary elements: original equipment manufacturers (OEMs), first-tier suppliers, sub-tier suppliers, raw and processed materials suppliers. However, individual companies may operate in several different positions in the supply chain. A company may work for many customers and functions as a first-tier supplier on one project and a sub tier supplier on other projects.

Members of the auto industry generally acknowledge that imperfect interoperability is an important and expensive problem. In order to resolve this problem, the SBM service platform is designed, and the construction of virtual enterprises is a prerequisite to use SBM service. All the stakeholders have to register to be users of the platform, if they want to use the interoperability service utility to support their business activities. After finishing the registration, the OEM users have authorization to create a virtual enterprise. They can choose appropriate suppliers from all the registered users to supply the parts/materials they need for normal production. After the OEM gathering all the suppliers it needs, a virtual enterprise is formally constructed. Then, master of the V.E. could apply for services such as SBM service and APO service. After been approved by the platform, the master could use SBM service with the other members of V.E. to establish interoperability easily and quickly.

It must be addressed that, there exists three different kinds of virtual enterprises in the platform, and the platform can serve all of them well. The first kind of V.E. is the most common and standard one. In this kind of V.E., an OEM is the unique master, and a number of suppliers serve it. The relationship among them is very simple because there is no intersection in the organization. However, in the second kind of V.E., the relationship is more complicated, because intersections appear among V.E.'s boundaries. One supplier may serve several masters. As a result, they will exchange information with different OEMs. The third kind of V.E., is the most complicated one. In the service platform, all the enterprise users could apply for SBM service, so it potentially allows a "double-role" situation. In other words, the master of a V.E., may be a supplier of another V.E.. Just take an engine manufacture enterprise for example, it can be master of a V.E., and get outsourcing parts to produce engines; meanwhile, it may play the role of a supplier to serve other OEMs. According to the analysis of these possible cases, we design and develop the service platform in a flexible way to deal with various possible requirements.

4. Enterprise Interoperability Service Platform

4.1. A Brief Introduction

SBM service platform provides a cooperative environment of supply business between OEMs and their suppliers. With the help of SBM, OEMs can share

production related information easily with their hundreds of suppliers. SBM provides several services to support the daily supply business activities, such as plan service, order service, finance service, quality service, notice service, inventory service etc.

Using the plan service, OEMs could publish their procurement plans to suppliers to guide their production and delivering of materials. There are mainly three kinds of purchase plans, including monthly plans, weekly plans, and daily plans.

Order service may be the most important function of SBM service. OEMs allocate orders to the suppliers according to the pre-defined quota standard. With the help of order service, the decomposed orders are published immediately to corresponding suppliers. Then, the suppliers will print orders online and ship the materials directly to OEMs' warehouses.

In the same way, finance service publishes all the financial information to suppliers, such as general ledger arrearages, quality compensation; inventory service allows suppliers to get the accurate inventory in assembly factory; quality service gives all the quality related data, such as quality inspection, sampling inspection, etc.

However, in order to serve all kinds of OEMs, the SBM service is designed and implemented flexibly, which provides the function of customization. That is to say, users of SBM could customize the service and use it in a DIY way. In order to solve the heterogeneous data integration problems, a data synchronization toolkit, one of the enabling services provided in SBM, is used to synchronize important data from inner systems of an OEM to the database of SBM service. The detailed structure of the synchronization toolkit will be introduced later.

4.2. Architecture Framework

Figure 2 shows the logical architectural framework of SBM service platform designed in SaaS model. This framework provides the capability of hosting multi-tenants using a single application instance supported by the same set of backend servers. In this framework, the presentation and customization modules of the application are deployed in web server, and the Security Service, Metadata Services and Business Services etc, are deployed in application server.

In the framework, business services play an important role in offering the basic functions mentioned above; presentation module deals with the user interface; with the help of metadata services, customization module allows the OEMs to configure the application according to their business needs; security services control the access to OEMs' business data and also ensure its security; shared services such as monitoring service and billing service are used by all the OEMs.

As is shown in Figure 2, a centralized UserInfor database is used to store all the information of users. That's to say, SBM service manages a central user account database that serves all of the application's end users. Each

OEM's administrator is granted permission to create, manage, and delete user accounts in the user account directory. In this way, there's no need to change the OEM's own user infrastructure. As a result, the OEMs could use SBM service more independently. Furthermore, in order to enhance security, besides the traditional username and password, an USB key is required when accessing to the SBM service.

As to the configurability, metadata is used to define all the variation points of SBM. Each of the OEMs has its corresponding metadata files, such as UI.xml, Role.xml, Menu.xml, DBconnection.xml, etc. Moreover, each OEM also has a dedicated metadata file folders to store these metadata files.

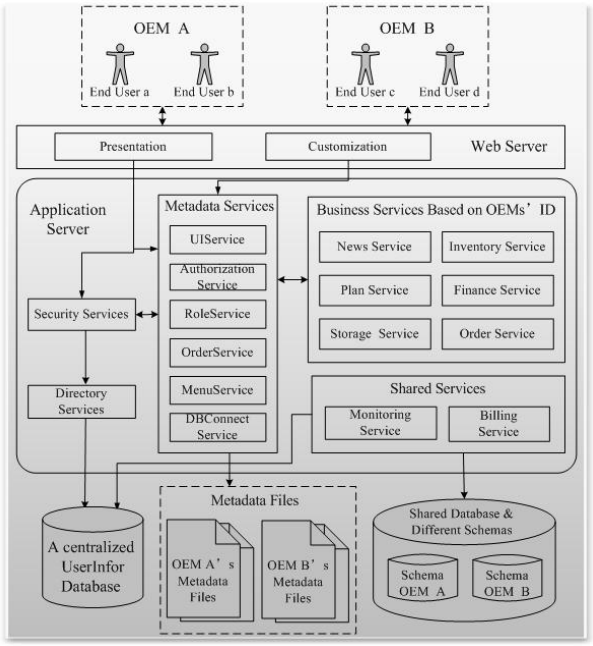

Fig. 2. Architecture framework of SBM service

In fact, metadata services are the most important part in a SaaS application, because: (a) all the variation points of SBM service are customized by OEMs throw metadata services; (b) nearly all the business services should use metadata services to parse corresponding metadata files, so as to perform in a desired way; (c) the changes of metadata should be hot deployed and be loaded into the application immediately by metadata services.

Generally speaking, a SaaS application primarily involves three types of data architecture [9]. You will find in Figure 2 that we use the shared database and different schemas to store OEMs' business data. Considering the hardware cost and the security of data, we finally choose this kind of data architecture. With the support of SBM service, assembly factory could

enhance and improve interoperability with its suppliers with little IT investment and operational cost, instead of developing a new solution from scratch.

4.3. JMX-based Metadata Management

As is mentioned above, all the variation points of SBM service are defined with metadata and stored in the form of XML. As a result, each OEM has a corresponding file folder to store its metadata files.

In order to realize the configurability of SBM, Metadata Services are used to parse these metadata files and load the customized information into the application. At present, there are mainly two ways to load metadata files: (a) load the metadata files when the application is initialized; (b) load the metadata files when they're required. Because almost all the functions of SBM service could be customized by OEMs, if the metadata files are loaded when they're required, the performance of SBM will be severely affected. Therefore, we load all the metadata files at startup, and all needed information is stored in memory, and the performance will be higher.

In practice, OEMs may often change some of the metadata files during runtime. Because this application is critical to the business, it needs to be configurable without causing a shutdown of operations. That's to say, you cannot restart the application to reload the changed metadata files, because multi-tenants are using the same instance. OEM A's customization of metadata files must be transparent to OEM B. A general way to solve this problem is to reread and reload the changed metadata files after they're modified and saved. The advantage of this way is that, you need not restart the application. However, the disadvantage is that, the changes of metadata cannot work immediately until the changed metadata files are reread and reloaded. What's more, even if a single line of the metadata file is changed, the whole metadata file must be reloaded. As a result, the users will get a bad performance.

In order to thoroughly solve this problem, we used the JMX[14] to realize the hot deployment of metadata. With the help of JMX, only the changed metadata will be reloaded immediately to the memory, and won't have to reload the whole metadata files.

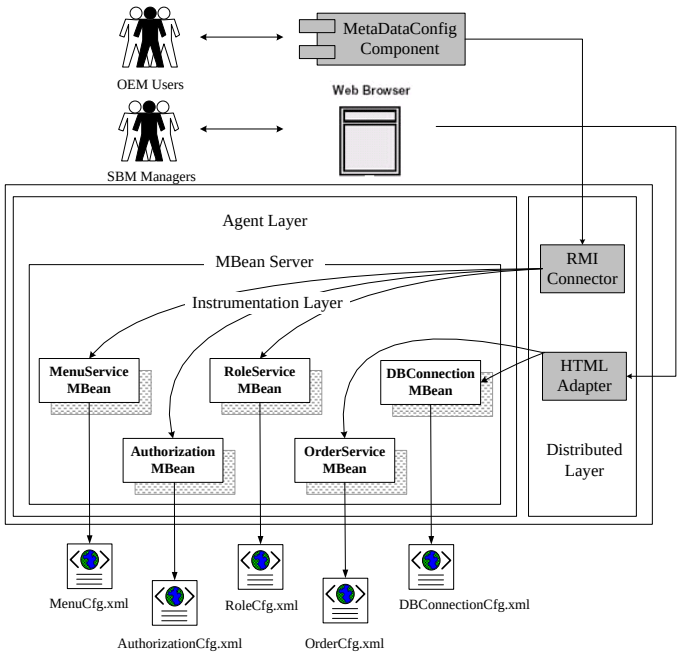

Fig. 3. JMX based metadata management

As is shown in Figure 3, the JMX architecture has three layers: instrumentation layer, agent layer, and distributed layer. JMX uses Java classes called managed beans (MBeans for short) to expose predefined portions of one application. In SBM service, every metadata file has a dedicated MBean defined and used to manage it. For example, the MenuServiceMBean is used to manage MenuCfg.xml. Furthermore, MenuServiceMBean can be defined to control all the elements and attributes of MenuCfg.xml, such as 'menu_id', 'menu_name', etc. Consequently, only the changed elements or attributes in MenuCfg.xml will be hot deployed directly into memory, the whole file won't be reloaded.

In JMX, all the MBeans have to be registered to the MBean Server in agent layer. Management tools access these MBeans by interacting with JMX agents that make the MBeans available to any number of protocols and technologies such as SNMP, Java RMI, and HTTP. It is clear that in Figure 3, some of the MBeans are used by the OEMs users to customize metadata files. These MBeans are connected with RMI connector by Metadata Configure Component. On the other hand, the rest of MBeans are used by the SBM managers to manage the runtime behavior, for instance, dynamically modifying the size of database connection pool. These kinds of MBeans are connected by HTML adapter, and can be managed directly through a web browser.

5. Scalable Deployment Architecture

A well designed SaaS application can be scalable to an arbitrarily large number of customers, and the number of servers and instances on the back end can be increased or decreased as necessary to match demand, without requiring additional re-architecting of the application, and all the changes can be rolled out to thousands of tenants as easily as a single tenant. Scalability is one of the key problems that need to be addressed. Generally speaking, the scalability of a SaaS application is often discussed in application layer and database layer.

5.1. Scalability in Application Layer

SBM service is designed to be scalable, fast, reliable and able to handle fail-over behavior. However, with an increased number of end users, the performance of SBM degrades and it is necessary to distribute client requests to different servers in order to perform parallel processing. Client requests towards web servers are distributed using a common front-end load balancer and high availability is achieved using a set of clustered servers with the same set of files and similar configuration.

In the construction of server cluster, the use of session must be solved. Generally speaking, two ways are widely used to deal with session: session replication and session sticky. Session reception is time-consuming and may block the network. On the other hand, session sticky is simple to implement, but it cannot solve the fail over problem.

Shared nothing architecture [16] (SN) is a distributed computing architecture in which each node is independent and self-sufficient, and there is no single point of contention across the system.

As is indicated in Figure 4, according to SN architecture, the SBM service is designed to run in a stateless fashion, all the session data is stored in a central session sever which is accessible to any application instance. In this way, each transaction can be handled by one instance as well as any other; all the instances are equal with each other. So if one of the instances is down, the requests of users will be forward to another instance and the process is transparent to users. Both the load balance and the fail over requirements are addressed.

5.2. Scalability in Data Architecture

Data architecture is an area in which the optimal degree of isolation for a SaaS application can vary significantly depending on technical and business considerations. [9] introduces three approaches to managing multi-tenant data: separate databases; shared database, separate schemas; and shared database, shared schema.

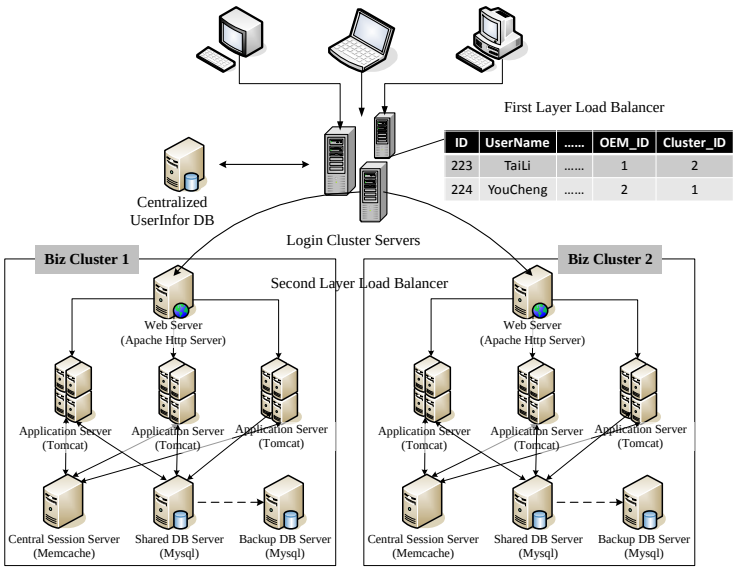

Fig.4. Business cluster based deployment architecture

Considering hardware and maintenance costs and data isolation requirements, the data architecture of SBM service follows the approach of shared database and separate schemas. The separate-schema approach is relatively easy to implement, and OEMs can extend the data model easily. This approach offers a moderate degree of logical data isolation for security-conscious OEMs, and can typically accommodate more OEMs at a lower cost.

As databases serve more users concurrently and grow in size, the amount of time it takes to perform operations such as querying and searching increases significantly. Therefore, a shared database should be scaled to accommodate more tenants or heavier usage when it can no longer meet baseline performance metrics. However, the separate-schema approach is well-suited to the tenant-based horizontal partitioning pattern because each OEM has its own set of data, so the managers can easily target individual tenant data and move it to another server. You can also find in Figure 4 that in order to ensure the high availability in data layer, we used a real time backup DB server to backup the production server.

5.3. Business Cluster-based Deployment Architecture

In general, in order to improve the scalability of a SaaS application, both the application server layer and the database layer should be scaled out. Assume that, if there are M application servers and N database servers in SBM service, the connections among them will be M multiple N. On the other hand, the database connections are very precious resources, when the operations

such as querying and searching increases significantly, the processing ability of the application will be in danger.

So we propose a *business cluster* based deployment architecture to solve this problem, as is indicated in Figure 4. In a business cluster, an application server cluster is bound with a concrete database server, and all the owners of the data stored in the database are bound with the application server cluster. As a result, an application server cluster is dedicated with a number of end users and their shared database server.

As is shown in Figure 4, requests from all the users are processed by the Login Server Cluster, which contains a centralized authentication system. In the UserInfor DB, each OEM is bound with a dedicated business cluster. As a result, all the requests from end users will be first dispatched to corresponding business clusters, so the Login Server Cluster works as the first layer loader balancer. However, within a business cluster, the requests will be dispatched by the web server which works as the second layer loader balancer.

As we can see from Figure 4, the SBM service can scale out almost infinitely simply by adding nodes in the form of inexpensive computers, since there is no single bottleneck to slow the system down. When the number of users is increasing rapidly, the deployment architecture can also meet the requirement of scalability. The business cluster based deployment architecture has been proved to be flexible and efficient.

6. Interoperability Challenges and Solutions

6.1. Integration Requirements

Most of the time, SaaS applications are only one small part of the overall IT-landscape of an enterprise. As far as it goes, most medium and large OEMs have certain applications already deployed on their premises. There's no question SaaS integration problems do exist: all the OEMs expect that SBM service can be integrated with their legacy applications. However, these kinds of integration requirements must be resolved; otherwise, the customers will leave and find another service provider. In the operation of our application, we also met the problem of integration. The integration requirements mainly exit in the process layer and data layer.

A business process supported by SBM service usually can trigger business process supported by another on-premise application. For example, an order process from SBM service would trigger an order fulfillment process managed by an ERP application. Therefore process integration can automate the end to end business process transaction spanned across SBM service and on-premise applications.

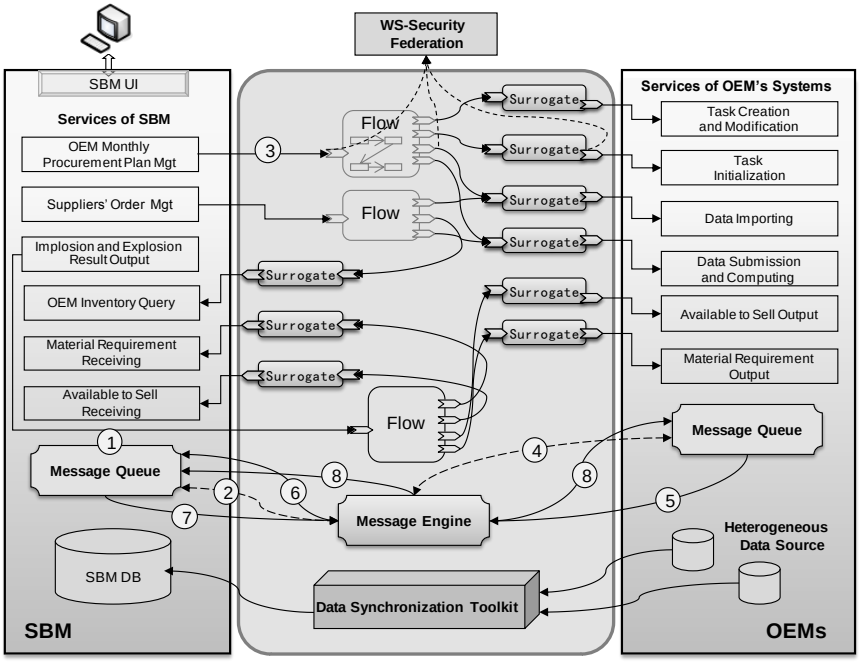

Fig.5. Integration roadmap between SBM service and on-premise application

There are two types of data in SBM service: master data and transactional data. These data should be synchronized from SBM services to OEM's on-premise applications and vice versa.

Figure 5 illustrates the integration roadmap we adopt to meet these integration requirements. In order to address the issues of data integration, we designed and developed a data synchronization toolkit to synchronize the important data between the SBM service and the heterogeneous data sources within OEMs. Furthermore, service surrogate extended from SCA is used to connect SBM service with the of inner systems, and in order to ensure the interoperability between them, a tool named Message Engine is used to provide coordination functions.

6.2. Data Synchronization Toolkit

As is shown in Figure 6, a data synchronization toolkit is used to synchronize the important data between the SBM service and the heterogeneous data sources. The synchronization toolkit has two parts: the client side is deployed within OEMs, and the server side is deployed in SBM service. The client side is used to extract the needed data from heterogeneous data sources and send them to the server side. On the other hand, the server side is used to receive and load the data into SBM service.

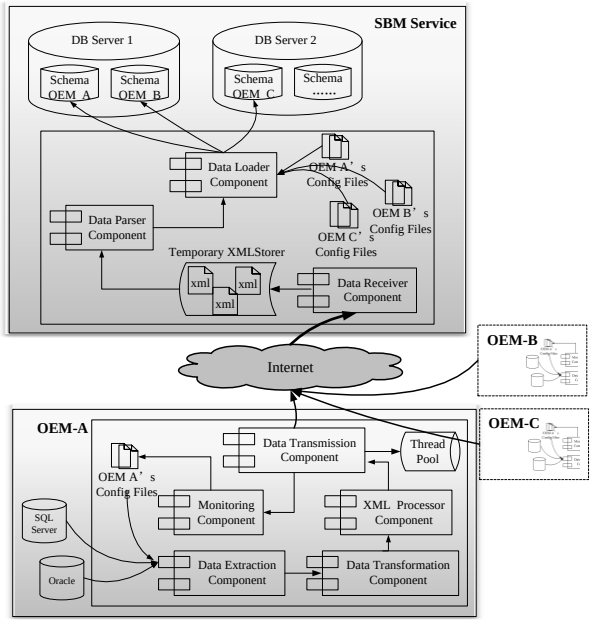

Fig.6. Data synchronization toolkit

It must be pointed out that, the data synchronization toolkit also works in a SaaS way. Because OEM's data may be stored in different databases and in different structures, the synchronization requirements must be different. So the data synchronization toolkit is designed and delivered with configurability: both the client and the server side provide a set of configure files which can be customized by OEMs according to their individual situation.

The client side deployed within OEMs will first extract the needed data from heterogeneous data sources according to the configure files. Then, all the data will be transformed into XML files. As a result, the difference among heterogeneous data sources is shielded. After that, a dedicated XML processor component is used to split and compress these XML files. Finally, the processed XML files will be transferred through internet.

On the server side, the received XML files will first be stored in a temporary file folder and then be parsed and loaded into dedicated schemas according to the configure files. The whole process will be monitored, in order to deal with some unpredictable situations.

6.3. Service Connection

In order to ensure the interoperability in process layer, several technological issues have to be addressed: (a) how to connect the web services exposed by SBM service and on-premise applications to execute business process and

get the needed data; (b) how to coordinate these services and make them cooperate correctly.

Generally speaking, there are various services in a heterogeneous environment; web service is just one of them. How to identify them and make them work together is an issue need to be addressed. This paper provides solutions including service authentication, service authorization and service connection according to different business requirements.

Service surrogate extended from SCA [18] provides a programming model which is used to construct applications and solutions based on SOA. SCA (Service Component Architecture) is a model that aims to encompass a wide range of technologies for service components and for the access methods used to connect them.

As to access methods, SCA compositions support various communication and service access technologies such as web service, messaging system and RPC (Remote Procedure Call). The original definition of SCA Component defines the implementation, exposure and invoking mechanism of services. However, it cannot fulfill our requirements for interoperability, such as service authorization and service authentication.

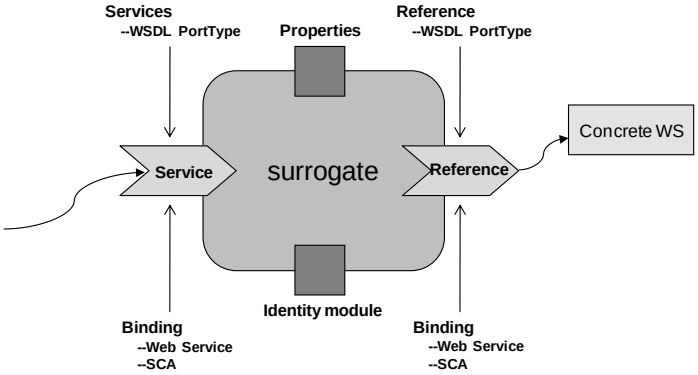

Fig.7. Definition of surrogate based on SCA component

SCA provides service flow definition mechanism to carry out complicated business processes automatically. It is the same as the extended SCA component. SCA describes the content and linkage of an application called composites. Composites can contain components, services, references, property declarations, plus the wiring that describes the connections among these elements. Composites can group and link components built from different implementation technologies, allowing appropriate technologies to be used for each business task [9].

This paper extends SCA Component by adding the Identity Module to monitor and process SOAP messages based on WS-Security. As is revealed in Figure 7, the extended SCA Component is named “surrogate”. In general, SBM service and on-premise applications manage their users separately. So the detailed solution of service connection is:

1) Encapsulating the related web services with SCA Component.
2) Process the authentication information in the identity module.

A mapping table for users should be maintained in SBM service. The table stores the mapping relationship of SBM users and on-premise application users. The role and authorization will not be changed and still be managed separately. When an authorized user, such as Bob logs in SBM service and invokes services provided by on-premise application, the authentication process will be activated. Firstly, Identity Module will insert Bob's identity information into SOAP head; then the gateway will capture the message and find the corresponding user of Bob in the mapping table. If succeeds, the new user_id certified will be inserted into SOAP head to invoke the services exposed by on-premise application. The return process is carried out in the same way.

6.4. Service Coordination

Service coordination is often used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed transactions. In order to ensure the interoperability between SBM service and on-premise application, this paper proposes a Message Notification based service coordination technology, which is driven by business rules. We design and implement a tool named Message Engine in dependence on Apache Kandula2 [17] project, which provides an open-source implementation of WS-Coordination, WS-Atomic Transaction, and WS-Business Activity.

The Message Engine not only enables SBM service to create a context needed to propagate an activity to on-premise application, but also enables the coordination of transactions among them.

As is shown in Figure 5, with the help of the Message Engine, the whole process of service coordination based on Message Notification is carried out by the following steps:

1) Configuring the Message Queue of SBM service and on-premise application;
2) Registering the Message Queue of SBM service to Message Engine;
3) Users log in SBM service and then activate a business process;
4) Web services involved in the process are invoked and Message Queue of on-premise application is registered to Message Engine;
5) Message Queue in on-premise application invokes the completed() function automatically after the execution;
6) Message Engine invokes the complete() function of SBM to notify the end of the process;
7) On receiving the notification, the whole process ends;
8) Notifying on-premise application the whole process ends.

With the help of these technologies, interoperability between SBM and on-premise application can be established to fulfill the requirement of making them work together.

7. Conclusions and Future Work

This paper proposed a methodology that provides a guide on how to establish interoperability between enterprises through a federated approach. Under the guide of this methodology, the paper designed and realized an interoperability service platform to enhance the interoperability and efficiency of the supply chain management in automobile industry. The architecture framework of SBM is introduced, and metadata is used to define all the variability points. In this paper, JMX is used to manage all the metadata files. In addition, shared nothing architecture is used to design the business cluster based deployment architecture. Data synchronization toolkit and message engine are implemented to address the integration issues in business process layer and data layer.

As far as it goes, the SBM service has been used by more than 1,200 suppliers, and the number is increasing. With the help of SBM service, interoperability between OEMs and suppliers is constructed and enhanced. On average, nearly 10,000 orders are transported by the data synchronization toolkit every day. However, there are also several problems need to be addressed in the future. For example, the data synchronization toolkit needs to be improved, because it'll be influenced easily by the condition of network. Meanwhile, we should do more research in the security aspect to ensure the security of tenant's data. Besides, more work will be done in the research of SLAs and QoS of services.

8. Acknowledgement

The authors would like to acknowledge the support provided for the project by the National Natural Science Foundation of China (60703027), the National High Technology Research and Development Program of China (2006AA01A113, 2009AA043506), the Science & Technology Development Projects of Shandong Province (2007GG10001001), the Natural Science Foundation of Shandong Province (ZR2009GM028), and the China's Post-Doctoral Science Fund (20090451313).

9. References

1. Yannis Charalabidis, George Gionis, Karl Moritz Hermann, Cristina Martinez; "Enterprise Interoperability Research Roadmap";Feb.2008
2. G arokh, M. J.; Ghahremanloo, Hoda; Karami, Mahnaz, Agility in Auto Dealers SCM, Service Operations and Logistics, and Informatics, 2007. SOLI 2007. IEEE International Conference on Volume, Issue , 27-29 Aug. 2007 Page(s):1 – 6
3. Ross, A. Creating agile supply chains, Manufacturing Engineer, Dec. 2003, Volume: 82, Issue: 6, pp: 18-21

4. Ralph Mietzner, Frank Leymann. Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-Tenancy Patterns. The Third International Conference on Internet and Web Applications and Services.
5. Abhijit Dubey and Dilip Wagle, “Delivering software as a service”, The McKinsey Quarterly, Web exclusive, May 2007
6. J. Bosch. Design and use of software architectures: adopting and evolving a product-line approach. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.
7. R. Mietzner and F. Leymann. Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors. In IEEE International Conference on Services Computing, 2008.
8. F.Chong and G. Carraro. Building Distributed applications: Architecture Strategies for Catching the Long Tail. MSDN architecture center.
9. F.Chong and G. Carraro. Building Distributed applications: Multi-Tenant Data Architecture. MSDN architecture center.
10. Thorsten Scheibler, Ralph Mietzner. EAI as a Service - Combining the Power of Executable EAI Patterns and SaaS. 12th International IEEE Enterprise Distributed Object Computing Conference
11. ISO 14258, Industrial Automation Systems—Concepts and Rules for Enterprise Models, ISO TC184/SC5/WG1, April 14, 1999.
12. M.Jaring and J. Bosch. Architecting product diversification- formalizing variability dependencies in software product family engineering. In QSIC ’04: Proceedings of the Quality Software, Fourth International Conference, pages 154–161,Washington, DC, USA, 2004. IEEE Computer Society.
13. D. Wang, K.L. Yung, W.H. Ip, “A heuristic genetic algorithm for subcontractor selection in a global manufacturing environment”, IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews, 2001, 31 (2), pp.189–198.
14. BEN G. SULLINS. JMX in Action. Manning Publications, ISBN 1-930110-56-1, 2003.
15. Thomas Kwok, Thao Nguyen. A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application. In IEEE International Conference on Services Computing, 2008.
16. http://en.wikipedia.org/wiki/Shared_nothing_architecture
17. http://ws.apache.org/kandula/2/index.html
18. SCA Service Component Architecture – Assembly ModelSpecification.http://www.osoa.org/display/Main/Service+Component+Architecture+Home

Yong Zhang has received his BS and MS degree in software engineering from software college, Shandong University, China. Currently, he is enrolled at the school of computer science and technology, also in Shandong University, as a PhD candidate working on services computing, grid computing. His current research interests include SOA, Web Service, grid computing and services oriented computing, enterprise interoperability.

Shijun Liu is an associate professor of the School of computer science and technology at Shandong University, China. He earned his BS degrees in Oceanography from Ocean University of China, and MS and PhD degree in computer science from Shandong University. His teaching and research interests are focused on Service Computing, Networked Manufacturing, Computer Integrated Manufacturing System and Manufacturing Grid. He has published 2 books and over 50 papers in journals and conferences and obtained many academic awards including the Nation Scientific and Technological Progress Award in 2002.

Lei Wu has received her Ph.D. degree in computer software and theory from Shandong University, China. Now, she is a lecturer at the School of Computer Science and Technology at Shandong University. Her research interests are focused on networked manufacturing; computer integrated manufacturing systems and manufacturing grids.

Yuchang Jiao has received his master degree in software and theory from Shandong University in 2008. He is currently working at IBM SOA Design Center, Beijing, China. His current research interests include Service Oriented Architecture, services computing and virtualization technologies.

Received: April 14, 2009; Accepted: September 26, 2009.