An Interoperability Points Based Interoperability Approach for SaaS Applications

Yanyan Han, Lei Wu, Shijun Liu, and Xiangxu Meng

School of Computer Science and Technology, Shandong University
250101, Jinan, P.R. China
sduhyy@foxmail.com, {i_lily,lsj,mxx}@sdu.edu.cn

Abstract. SaaS applications have been widely adopted especially by small and medium enterprises. At the same time, the features "multi-tenancy" and "loosely coupled" bring new challenges to enterprises interoperability. On the basis of the layered interoperability model, the paper presents an approach based on interoperability points to implement interoperation between SaaS applications in the service layer. After carrying out the interoperability point matching algorithm, the intermediary Enterprise Service Bus (ESB) performs dynamic selection of interoperability points dictated by Quality of Service (QoS) attributes. In the premise of a comprehensive consideration of the functional and non-functional preferences and constraints, dynamic interoperation between SaaS applications is realized. Finally, this paper shows a case study of applying the interoperability approach.

Keywords: enterprise interoperability, SaaS, interoperability point.

1 Introduction

In the current industrial and economic context, enterprises should be capable of seamlessly interoperating with other enterprises across organizational boundaries to gain more benefits. Enterprise Interoperability (EI) has therefore become an important area of research to ensure the competitiveness and growth of enterprises [1].

At the same time, SaaS (Software as a Service) [2] has been widely accepted as a popular way to carry out the software service delivery. SaaS applications have been adopted by more and more business partners, especially by small and medium enterprises (SMEs). Software delivered in a SaaS model is no longer running exclusively for one customer at a customer's premise but supports multi-tenants over the Internet, which is called "multi-tenancy". Enterprises once accomplish their business through the interaction between traditional on-premise software must face with the interoperability issues between SaaS applications hosting anywhere. The feature of "loosely coupled" means that interoperability bridge between two SaaS applications must be services with standard interfaces. The above two features are exactly two main challenges of interoperability between SaaS applications [3].

In this paper, we focus especially on the new framework and approach to implement interoperation between SaaS applications in the service layer. In our

M. van Sinderen et al. (Eds.): IWEI 2013, LNBIP 144, pp. 22–35, 2013.

proposed framework, a SaaS application which wants to interoperate with other SaaS applications should expose a standardized web service interface as an interoperability point which acts as a source interoperability point. After searching among other interoperability points according to the basic attribute constraints analyzed from the interoperation request, we can gain several related interoperability points. On a basis of an interoperability point matching strategy, we put forward an interoperability point matching algorithm. The algorithm takes the operation interfaces of the related interoperability points as input, and produces some target interoperability points sorted by matching degree. The intermediary ESB performs dynamic selection of these target interoperability points dictated by QoS attributes and gains the optimum interoperability point to interoperate with. The dynamic interoperation between SaaS applications is realized finally.

The following parts of the paper are organized as follows: Section 2 introduces the previous work on the layered interoperability model as well as the research actualities of enterprise interoperability. In section 3, we present an overview of the interoperability framework in the service layer and the main components, which is followed by section 4 that describes the process of interoperability point discovery. Section 5 discusses the implement of dynamic interoperation based on ESB. Section 6 presents a case study. Finally, conclusions and future work directions are shown in the last section.

2 Related Work

Researchers have presented many initiatives which are concerned with the elaboration of an enterprise interoperability framework. Kassel [4] presents some foundations for introducing a decision support model into a model-driven interoperability architecture for services. Arafa et al. [5] set out a framework for a high-level approach to software component integration. For another work, Yang et al. [6] provide a novel service and data management platform called DSP (Data Service Portal) that facilitates the integration of applications by sharing their information in a loosely coupled manner. Other significant pieces of work such as the LISI approach [7], the IDEAS interoperability framework [8] and the European Interoperability Framework (EIF) [9] aim at different concerns.

In previous work, we have designed an approach to develop SaaS applications and implemented a service based collaboration supporting platform (New Utility platform & Tools for Service, Nuts) [10] to deliver them to enterprises. With the ultimate aim to provide means to resolve all kinds of interoperability challenges that may hamper the effective usage of SaaS applications in supporting enterprise collaborations, we have given the definition of the "layered interoperability model" [11].

For the interoperability of independent SaaS applications must be implemented from the UI layer to the data layer underneath. The enterprise interoperability framework is designed as a layered model with 5 layers including data layer, service layer, process layer, business layer and presentation layer.

A modified Widget model is used to implement interoperation in the presentation layer. Service layer interoperability refers to discover, composite different kinds of

application functions or services for well collaborative work, which is the core interoperability of the five layers. The goal of interoperability in the process layer is to make various processes work together. Interoperability for the business layer is on the standpoint of organization and company, and it deals with the interoperation barriers causing by diverse business rules, policies, strategies, legislation and culture. Business layer interoperability is established by negotiation mechanism and monitoring facilities, which makes the use of a federated analogous interoperability form. Data synchronization toolkit and message engine are implemented to address the integration issues in data layer.

The interoperability in the same layers is interconnected by two or several interoperability points. The interoperability point is defined as an interface between two interoperability entities and has different forms in different interoperability layers. To implement interoperation between two SaaS applications, we should detect and define the interoperability point for different SaaS applications in different layers. Focusing on the interoperability approach for SaaS applications in the service layer, this paper outlines an interoperability framework and gives the formal definition of the interoperability point in the service layer.

3 The Interoperability Framework in the Service Layer

In the subsections below, we give a brief overview of the key components in the framework as shown in figure 1.

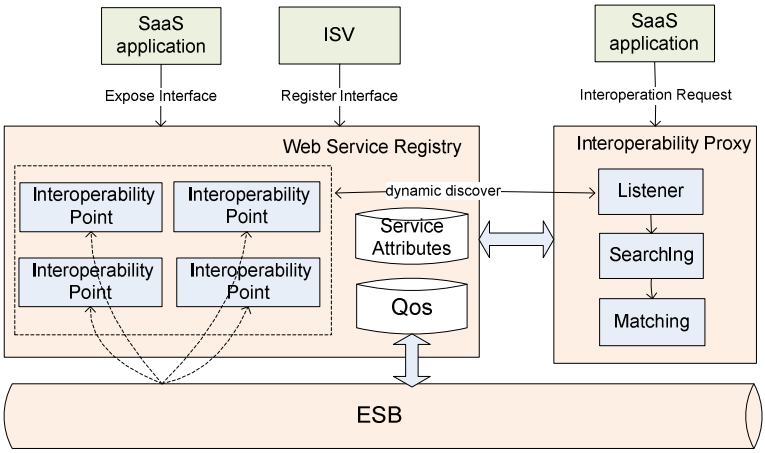

Fig. 1. The interoperability framework in the service layer

3.1 Web Service Registry

This module mainly includes two components:

1. Register Interface

Web Service (WS) technologies rapidly become the de facto standard to expose the functions of business application. The ISVs (independent software vendors) package

and publish the business functional modules as web services ahead of registering to the web service registry on the platform. The platform automatically extracts the service metadata information from the given WSDL, such as service name, operations, input/output parameters, etc. On this basis, the ISVs need to add some other service attributes to complete the registration, such as service description and service type.

Our Web Service formalization definition is given below:

Definition 1 (Web Service): A Web Service is a tuple WS = (SA, OPs), where: SA delegates the public attributes of service, including service name, service description and service type; OPs is a finite set of operations, then for every OP ∈ OPs, OP = (Oname,Ins,Os), Oname is the operation name; Ins represents the input parameters of OP; Os represents OP's output parameters, for each I ∈ Ins, O ∈ Os, I = (Ipname,Iptype), O = (Opname,Optype). Ipname and Opname are respectively the input parameter name and the output parameter name. Iptype and Optype are the input parameter type and output parameter type.

The web service registry realizes the service classification, the standards and specifications of the service description and enhanced service discoverability. Through the unified classified standard, services that registered in web service registry can be searched by name or other constraints.

2. Expose Interface

The interoperability point is the interface between two interoperability entities, namely two SaaS applications. In order to fully realize interoperation among SaaS applications, we should define interoperability points formally and analyze the procedure of searching, matching and selection of interoperability points.

The SaaS application can selectively expose the registered web service as an interoperability point. Only by the exposure operation will the SaaS application be possible to interoperate with other SaaS applications. The interoperability point not only inherits all the attributes of the web service, but also appends several new attributes, such as enterprise attributes, QoS attributes and URI. The enterprise attributes can be used as one of the conditions of interoperability point searching. By identifying the enterprise attributes, SaaS applications can interoperate with related enterprise's SaaS applications. Interoperation between SaaS applications should be based on mutual trust. In some cases, SaaS applications only hope to interoperate with their partner enterprises' SaaS applications. The QoS attributes can be updated by the monitor on the Nuts platform in real-time and can be used as the basis for the ESB-based dynamic selection among target interoperability points. The URI uniquely identifies the interoperability point which serves as the entry point for the interoperation call.

The formalization definition of Interoperability Point:

Definition 2 (Interoperability Point): A Interoperability Point is a tuple IP = (SA, OPs, QoS, EA, URI). It inherits the whole attributes of Web Service. QoS, EA and URI are three new attributes, where: QoS attributes including response time, reliability and usability; EA means the enterprise attributes; URI uniquely identifies an interoperability point and serves as the entry point for the interoperation call.

3.2 Interoperability Proxy

The interoperability proxy is responsible for interoperability point discovery. Similar with web service discovery, the interoperability point discovery in this paper refers to obtaining target interoperability points which both satisfy the users' basic attribute constraints and match with the source interoperability point according to the operation interface constraints.

The proxy briefly includes several following components:

1. Listener Component

1) Listening interoperation request

This component carries on the analysis of the interoperation request and obtains the basic attribute constraints of interoperability points, such as service name, service type, enterprise attributes and so on.

The formalization definition of Interoperation Request:

Definition 3 (Interoperation Request): IR = (SN, SD, ST, EA, w), where: SN: service name; SD: service description; ST: service type; EA: enterprise attributes; w: the threshold value of matching degree between interoperability points.

2) Listening fresh exposure of interoperability points

The framework is also able to support run-time interoperability point discovery. The listener component can dynamically discover new interoperability points exposed by SaaS applications. According to the current interoperation request, it determines whether the new interoperability points can be used as new target interoperability points.

2. Searching Component

In a large scale of interoperability points, how to discover the target interoperability points rapidly, accurately and efficiently is a tough problem. In order to reduce the time consuming of the interoperability point matching algorithm, we divide the process of interoperability point discovery into two phases, namely the searching phase and the matching phase. In the searching phase, the proxy obtains several related interoperability points after querying according to the basic attribute constraints in the interoperation request. An operation interface matching algorithm is applied to related interoperability points in the next step. This strategy can effectively filter out the irrelevant interoperability points, reduce the input range of the matching algorithm and improve the efficiency of the algorithm.

3. Matching Component

To enable interoperability points seamlessly interact with each other, the way how to design the interface matching algorithm is a key. We put forward a matching algorithm for the operation interfaces of interoperability points. On the basis of related interoperability points get from the last searching phase, we can get a set of target interoperability points ranked according to the matching degree. A number of different business processes will be formed after invoking the matching algorithm.

The same web service exposed by different SaaS applications may become different interoperability points which have the same web service attributes. For example, the interoperability points IP5, IP6 and IP7 in the figure 2 are exposed from the same web service WS2, but they belong to different SaaS applications.

At the same time, the same SaaS application may deploy multiple instances, so there may also exist interoperability points possessing the same web service attributes. For example, the interoperability points IP1, IP2 and IP3 in the figure 2 which belong to the different instances of the same SaaS application also possess the same web service attributes.

The target interoperability points which have the same web service attributes possess the same matching degree after matching with the source interoperability point, so the searching and matching process can be omitted. Meanwhile, they generate the same business process, the user can choose according to their actual needs as well as the matching degree obtained. As shown in figure 2, after interoperability point searching and matching, two processes have been generated: IP0—>{IP1, IP2, IP3, IP4}; IP0—>{IP5, IP6, IP7}.

After performing the selected process, ESB perform dynamic selection of these target interoperability points dictated by QoS attributes.

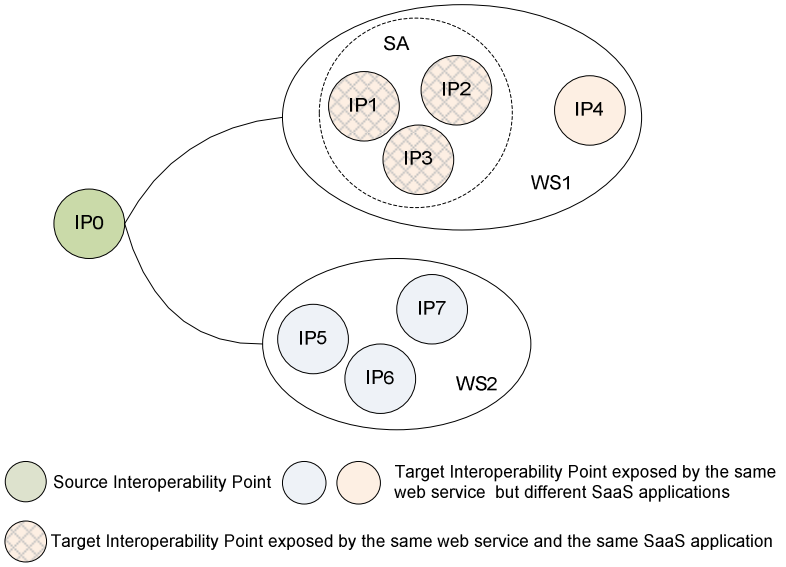

Fig. 2. The target interoperability point in a different case

3.3 ESB Routing Engine

Through the searching and matching performed by the interoperability proxy, we have get some target interoperability points which meet the goal of a business process. From the above, we know that there may be multiple target interoperability points, and new interoperability points that meet the request and matching rules may

be exposed, and some target interoperability points may be no longer available or can no longer respond to the request. In these cases, to obtain a fast response and high quality service, we need an intermediary to conduct the dynamic selection and the discovery of target interoperability points.

ESB is the core and basis of SOA, and one of its core functions is message routing [12]. Message routing mainly refers to the delivery of messages between request endpoint and provider endpoint according to certain rules and logic. In addition, ESB supports transport protocol conversion and message format conversion and applications are able to flexibly connect with each other, regardless of the platform and technical differences.

Consequently, we use ESB in our framework to determine an optimum interoperability point from candidates based on the QoS attributes in that we think the quality of the target interoperability point is one of the main concerns.

3.4 The Process

Using the interoperability framework, the process could be illustrated as follows:

1. The ISVs package the business functional modules in the SaaS applications, publish as web services according to defined rules and norms and register to the web service registry on the platform after determining some service attributes.
2. SaaS applications can selectively expose the registered web service as an interoperability point. Only by the exposure operation will the SaaS application be possible to interoperate with other SaaS applications.
3. The interoperability proxy obtains several related interoperability points after querying according to the basic attribute constraints in the interoperation request. On this basis, according to the operation interface matching rules, some target interoperability points and several business processes will be gained after invoking the matching algorithm.
4. After the searching and matching phase, ESB performs dynamic selection of these target interoperability points dictated by QoS attributes and obtain the optimum interoperability point ultimately.

4 Interoperability Point Discovery

Web service discovery is based on web service matching. The functionality provided by web services is accomplished by calling the operations. The operation is the basic functional entity of web services. Every web service comprises a number of operations. Service matching is ultimately reflected in the operation matching.

We have acquired a set of related interoperability points after the searching phase. In order to further find the target interoperability points that can actually interact with the source interoperability point, we make full use of the service operation structure information provided by the current standard service description language WSDL,

establish matching rules and design the interoperability point matching algorithm based on operation interface descriptions.

4.1 Interoperability Point Matching Rules

Each web service has an associated WSDL document, describing the service functionality and interface. Every service contains a series of operations and each operation is a set of names corresponding to the operation's input and output parameters. WSDL document describes the name and data type of each parameter in more detail.

The main content of the WSDL description document of a web service can form a tree structure logically. As shown in Figure 3, there are four layers in the figure. The root node represents an interoperability point. The nodes in layer 2 represent the operations. The nodes in layer 3 represent the input or output messages. And the nodes in layer 4 represent the parameters of the messages.

The input and output parameter types of operations are defined with XML Schema. The parameter type can be divided into simple data type and complex data type. Simple data type needs only parameter name and internally defined parameter type such as int and string. Each parameter is presented in the form of <name, type>. But for complex data type, the model group tags which nest other simple data types or complex data types are used.

We begin with the input and output parameters of operations and match them in three aspects: the number of parameters, parameter name and parameter type.

When the matching degree calculated by matching the output parameters of a source interoperability point and the input parameters of a related interoperability point in the aforementioned three aspects reaches the threshold user preset, the two interoperability points match successfully. And the related interoperability point can be treated as a target interoperability point.

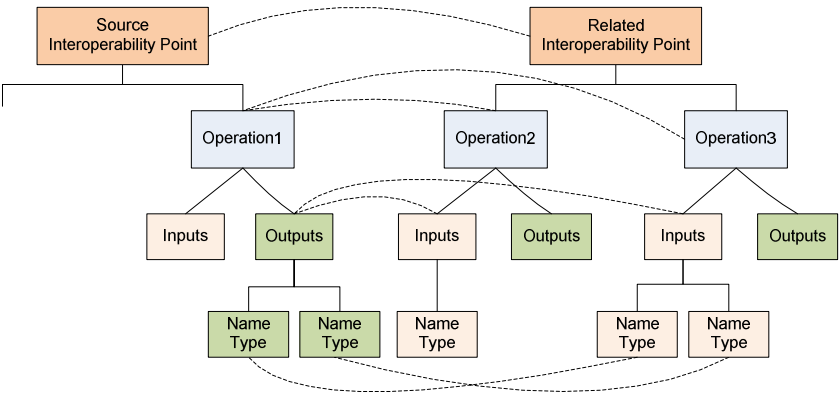

Fig. 3. Matching between two interoperability points based on the operation interface descriptions in the WSDL document

The concrete matching rules are shown as follows:

1) The number of the output parameters of the source interoperability point has to be the same with that of the input parameters of the related interoperability points. That is the precondition of the following matching processes.

2) The simple data type parameters are shown in the form of <name, type>, so matching degree is the combination of parameter names matching degree and parameter types matching degree. For parameter names, we can match them according to semantic similarity. For example, we can use the existing WordNet [14] semantic dictionary. For parameter types, we can reference the classification method in article [15].

3) The complex data type parameters nest other simple or complex data type parameters. So we implement algorithm with recursive.

4.2 Interoperability Point Matching Algorithm

The following are the main matching algorithms.

```
Algorithm 1 getTargetIPs
Input: SIP, the source interoperability point
       RIPs, the set of related interoperability
       points
       W, the threshold value of matching degree
Output: TIPs, the set of target interoperability points
Set sp as the operation of the source
interoperability point;
Set OPs as the set of target operations in target
interoperability points;
Set MD=0;
For each interoperability points RIP in RIPs{
   For each operation p∈ RIP{
   MD= getMatchDegree(sp,p);
     If(MD>w){
        RIP.OPs.add(p);
        If(RIP is not in TIPS)
           TIPs.add(RIP);
   }
}
```

Algorithm 1 matches the operations of source interoperability point with all the operations of the related interoperability points. The interoperability point whose calculated matching degree is greater than the threshold value user preset will be added to the set of target interoperability points.

Algorithm 1 calls Algorithm 2 to calculate the matching degree between operations.

```
Algorithm 2  getMatchDegree
Input:  sp, the source operation
         P, the target operation
0utput: MD, the matching degree between two
        operations
If(|sp.Os|==|p.Is|){
```

```
      For each parameter pairs{
      If(isSimpleType(sp.o)&&SimpleType(p.i)){
         nameMD=getNameMD(sp.o.name,p.i.name) ;
         typeMD=getTypeMD(sp.o.type,sp.i.type) ;
         MD=getMD(nameMD,typeMD)
     }
     Else if(isComplexType(sp.o)&&ComplexType(p.i)){
         If(|sp.o.groupLength|==|p.i.groupLength|){
             For each parameter pairs in model group
            getMatchDegree(sp.o.groupi,p.i.groupi);
          }
         ELSE MD=0;
      }
      Else MD=0;
      }
}
Else  MD=0;
```

Algorithm 2 is used to calculate the matching degree between the operations of interoperability points. Firstly it judges whether the number of parameters are the same. Secondly, the operation matching degrees of simple data type and complex data type are calculated respectively. The returned value is used in Algorithm 1.

The calculation of the matching degree of parameter names and parameter types is not the emphasis in the paper and no more words about it here.

5 ESB-Based Dynamic Interoperability

On the basis of functional matching, target interoperability points must guarantee some kind of quality. So we use QoS attributes information, such as response time, reliability and usability, as the basis of dynamic target interoperability points selection. NUTs platform provides a monitor which can update the QoS attributes information of interoperability points in real-time and the monitor can select the interoperability point with optimum performance according to some certain rules.

We need an intermediary to receive the request messages and route the messages to the target interoperability points. ESB implements message routing that receives and dispatches messages from source to the target. In addition, ESB establishes transport protocol conversion and message format transformation. Among several ESB implementations, we choose Mule and integrate it to our interoperability framework to realize the interoperation between SaaS applications.

Web Service Proxy is one of the commonest scenarios in ESB and also one of the four pattens of Mule.

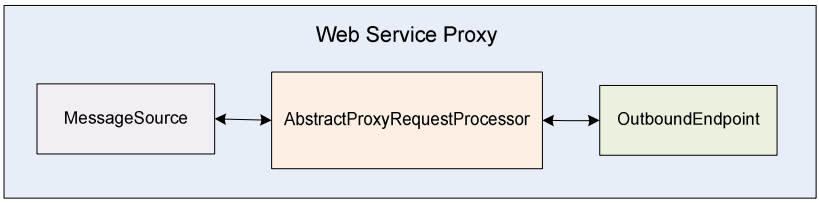

Fig. 4. The Web Service Proxy Pattern of Mule

There are three components in the Web Service Proxy, as shown in Figure 4.

1. MessageSource
MuleMessage is received or created by MessageListener. For example, If the DefaultInboundEndpoint is adopted as the MessageSource, SOAP messages will be received from the socket.

2. OutboundEndpoint
It is in charge of receiving and distributing messages.

3. AbstractProxyRequestProcessor
It is responsible for handing MuleEvent and rewriting WSDL addresses. There are two implementation classes, which are StaticWsdlProxyRequestProcessor and DynamicWsdlProxyRequestProcessor respectively.

By the following codes, we get the optimum interoperability point's address based on QoS analyzation and add an output endpoint with the new address dynamically. Then Mule can transfer the request messages to the optimum interoperability point.

```
//Clone a Global service
EndpointBuilder endpointBuilder =
muleContext.getRegistry().lookupEndpointBuilder("
originBuilder");
EndpointBuilder cloneEndpoint = (EndpointBuilder)
endpointBuilder.clone();
//Get the uri of optimum interoperability point
  from the QoS analyser
String uri=getUri(TIPs);
cloneEndpoint.setURIBuilder(new URIBuilder(uri));
//Rewrite the info for clone endpoint
muleContext.getRegistry().registerEndpointBuilde
r("optimumUri", cloneEndpoint);
//Get the OutboundRouter, clear the message and
add the new endpoint
OutboundRouter outboundRouter = ((OutboundRouter)
service.getOutboundRouter().getRouters().get(0));
outboundRouter.getEndpoints().clear();
outboundRouter.addEndpoint(cloneEndpoint.buildOu
tboundEndpoint());
```

6 A Case Study

This section demonstrates the features of our interoperability framework by referring to an example. On the NUTs platform, there exists a good deal of SaaS applications. Many SaaS applications expose the standardized web service interfaces uniformly registered by ISVs as interoperability points.

For Example, there are two SaaS applications on the delivery platform, one is supply business management system (SBM) and the other is Advanced Plan Optimization (APO). Several organizations tenant these SaaS applications and maintain their own instances. We can observe from figure 4 that SBM_A, APO_B and APO_C are three typical SaaS applications which expose some web service interfaces as interoperability points. If Tenant A which rents SBM_A wants to optimize the result plans list queried by PurchasePlanQuery, it can put forward an interoperation request. PurchasePlanQueryA should be treated as a source interoperability point and three target interoperability points will be figured out after the searching and matching process. Two different business process "Purchase Plan Query—>Supply Forecast" and "Purchase Plan Query—>Plan Optomize" will be presented to Tenant A.

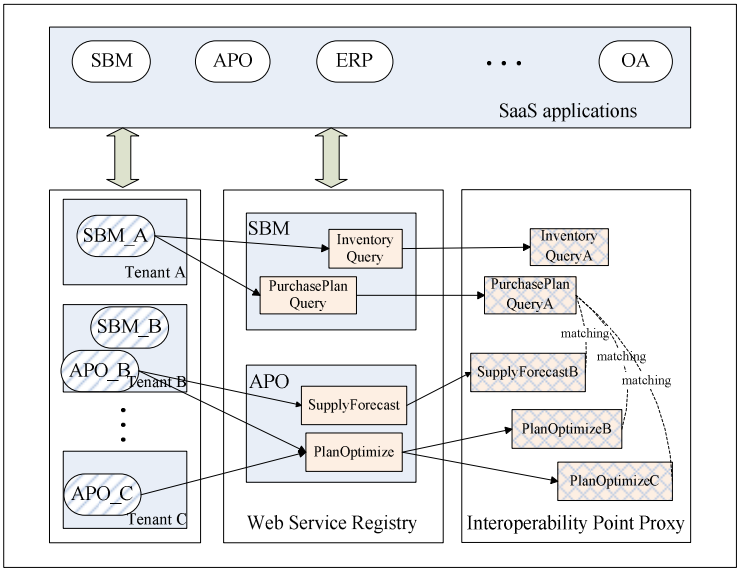

Fig. 5. An example process of interoperability point discovery

Tenant A should choose one of the business processes based on own preferences. Then ESB will dynamically select interoperability points and perform transport protocol conversion and message format transformation simultaneously.

The Inbound which serves as the request client of Mule receives the request messages. Web Service Proxy receives not only request message but also the optimum interoperability point address selected by the monitor on Nuts platform.

Web Service Proxy creates a dynamic endpoint and rewrite the OutboundAddress as the new endpoint address. When the call is triggered, Mule will deliver the request message to the optimum interoperability point and the dynamic interoperation between two SaaS applications is realized finally.

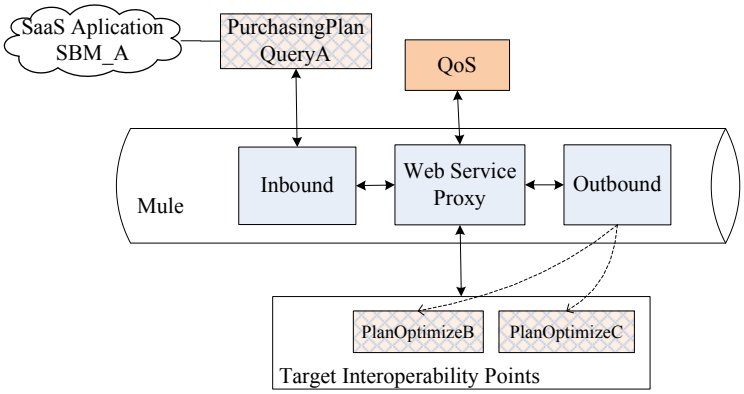

Fig. 6. An example process of dynamic interoperation between two SaaS Applications

7 Conclusion and Future Work

The paper presents an approach to implement interoperation between SaaS applications in the service layer. We provide the formalization description of the interoperability point and put forward an interoperability point matching algorithm on a basis of an interoperability point matching strategy. After interoperability point matching, the intermediary ESB performs dynamic selection of interoperability points dictated by QoS attributes. In the premise of a comprehensive consideration of the functional and non-functional preferences and constraints, we finally realize dynamic interoperation between SaaS applications.

In our algorithm, interoperability points are sorted in a particular order. We need match each interoperability point with the source interoperability point one by one exhaustively. The matching algorithm will meet efficiency problem when the number of interoperability points reaches some order of magnitudes. In our future job, index mechanism will be introduced to build the function index of interoperability points and a matching algorithm based on index will be provided.

Acknowledgment. The authors would like to acknowledge the support provided by the National High Technology Research and Development Program of China (2011AA040603, 2012AA040904), the National Key Technologies R&D Program of China (2012BAF12B07), the Natural Science Foundation of Shandong Province (ZR2009GM028, ZR2011FQ031) and Independent Innovation Foundation of Shandong University (IIFSDU).

References

1. Charalabidis, Y., Gionis, G., Hermann, K.M., Martinez, C.: Enterprise Interoperability Research Roadmap (2008)
2. Dubey, A., Wagle, D.: Delivering Software as a Service. In: The McKinsey Quarterly, Web Exclusive (2007)
3. Liu, S., Wang, L., Meng, X., Wu, L.: Dynamic Interoperability Between Multi-Tenant SaaS Applications. In: Enterprise Interoperability V. Proceedings of the IESA Conferences, vol. 5, Part 4, pp. 217–226 (2012)
4. Kassel, S.: An Architectural Approach for Service Interoperability. In: International Conference on Interoperability for Enterprise Software and Applications China, IESA 2009, pp. 212–218 (2009)
5. Arafa, Y., Boldyreff, C., Tawil, A., Liu, H.: A High Level Service-Based Approach to Software Component Integration. In: Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (2012)
6. Yang, J., Anand, R., et al.: Data Service Portal for Application Integration in Cloud Computing. In: Emerging Technologies for a Smarter World, CEWIT (2011)
7. The C4ISR Architecture Working Group (AWG) (CAWG): Levels of Information Systems Interoperability, LISI (1998)
8. IDEAS: Interoperability Development for Enterprise Application and Software Roadmaps, Annex 1—Description of Work (2002)
9. EIF: European Interoperability Framework, White Paper. Brussels (2004)
10. Nuts, `http://www.nutsplatform.cn`
11. Liu, C., Yang, C., Liu, S., Wu, L., Meng, X.: A Process Interoperability Method for SMEs. In: van Sinderen, M., Johnson, P. (eds.) IWEI 2011. LNBIP, vol. 76, pp. 50–60. Springer, Heidelberg (2011)
12. Keen, M., Acharya, A., Bishop, S., et al.: Patterns: Implementing an SOA Using an Enterprise Service Bus, Redbooks. IBM Press (2004)
13. Mule, `http://www.mulesoft.org`
14. Miller, G.: WordNet: A lexical database for english. Communications of the ACM 38(11) (1995)
15. Yu, S., He, F., Le, J.: Automatic Web Service Composition Based on Interface Matching. Computer Science (2007)
16. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
17. Zhou, C., Zhang, X.: A Policy-Confgurable Dynamic Routing Mechanism in Enterprise Service Bus. In: International Conference on Educational and Information Technology, ICEIT, pp. 480–485 (2010)
18. Ziyaeva, G., Choi, E., Min, D.: Content-based intelligent routing and message processing in Enterprise Service Bus. In: Proceedings of the International Conference on Convergence and Hybrid Information Technology, pp. 245–249 (2008)