Personalized Modeling for SaaS Based on Extended WSCL

Liu Ying[1,2], Zhang Bin[1], Liu Guoqi[2], Wang Deshuai[1], Zhang Yichuan[1]
[1](College of Information Science and Engineering, Computer Applications, Liaoning, Shenyang, 110819)
[2](Software College, Software Engineering, Liaoning, Shenyang, 110819)
{liuying, zhangbin}@mail.neu.edu.cn, liugq.neu@gmail.com, {sunwonders, zhangyichuan_33@163.com}@126.com

***Abstract*— Software as a service (SaaS) is an emerging software framework in which business data and logic typically integrate with other applications. It requires a unified subscriber to describe SaaS to make for easy integration; however, SaaS provides services to different tenants by running only one instance. In order to satisfy personalized needs from different tenants, the business logic becomes correspondingly complex. As this logic is cumbersome to reveal to every individual tenant, we propose the use of Web Services Conversation Language (WSCL) to express the views of tenant and provider separately. To overcome deficiencies in WCSL for expressing heterogeneous data, process rules, and business rules, we extend the syntax of WSCL. We also put forward a new modeling method for constructing SaaS Service, describing the modeling process and the algorithm for obtaining the tenant model from the business model. In conclusion, we describe the modeling tools and validation methods.**

Keywords-SaaS Service; ex_WSCL; Multi-Tenant; Business Model; Tenant Model

I INTRODUCTION

Software as a Service (SaaS) is a new emerging software design, implement and deploy model[1]. Software providers own the software and release its functions to a set of customers on the Internet. Software customers do not own the software themselves, but rather utilize it through an Application Programming Interface (API) accessible over the Web. Therefore, software is provided to the customer as a service rather than as a product. SaaS benefits from a standardized API to enable convenient interaction between software provider and customer. Web service technologies are fundamentally based on a set of Extensible Markup Language (XML) standards, such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI). One can use these standards in a Service Oriented Architecture (SOA) to describe Web services, communicate by message, and register services. In SOA[2], each service makes its functionality available through well-defined or standardized XML-format APIs, and supports interoperable application-to-application interaction over the Web. XML plays an important role in the data transport protocol for these technologies, with Web service technologies becoming widely deployed to implement SaaS[3]. In this paper, we consider SaaS implemented by Web service technologies as SaaS Service.

SaaS applications contain business data and logic which are usually required to integrate with other applications[4]. It needs a unified subscriber to describe SaaS to make its integration easy. The service provider delivers software functionalities with one single instance software application running for all of its tenants[5]. This one instance is used by different tenants having different personalized needs in terms of data, process rules, and business rules. Modeling a SaaS application based on service description is difficult when multiple parties are interacting. WSDL[6] provides a functionality interface, but it does not provide a way to describe business logic, and thus cannot satisfy the integration needs. Business Process Engine Language[7] (BPEL) models business processes from both control logic and data views based on the theory of workflow. The interaction is described at message level, but only represents control from one party's perspective. As there is only one instance running in a SaaS application, it should satisfy all possible personalized needs such as heterogeneous data, process rules, and business rules. Such a SaaS application model may be very complex, and if every tenant must include their business logic in this model, it will be difficult to keep their logic confidential. Consequently, we need a conversation model to track the message sequences among multiple parties and sources, and allow each involved party to describe its part in the interaction. Then we can define each tenant model only through its own business logic.

This work is supported by the National High-Tech research and Development Plan of China under Grant No.2009AA01Z122.

Fortunately, Web Services Conversation Language[8] (WSCL) fits the requirements above. WSCL uses conversation to describe business logic and abstract interfaces, and defines the public message exchanges that occur between interactive parties, rather than a specific business process that a single part executes. Conversation contains both elements of interaction and transition. Interactions model the actions of the conversation as documented exchanges between two participants. When two participants interact with each other, they perform corresponding operations. Transitions specify the ordering relationships between interactions to express business logic. When using WSCL to model a SaaS application, we can consider views of both SaaS application provider and tenants, and define the models as business models and tenant models. Business models define all of the tenants' business logic from the provider view, while each tenant model defines separate business logic only from each individual tenant view. When we use WSCL to model the interaction in a SaaS application, then each part involved can see the business logic from his own view. As each tenant knows only their own business logic, WSCL also prevents business security from being compromised between tenants.

WSCL provides a conversation to conveniently describe the coordination between SaaS provider and tenants. However, when a business model is described from the provider's view, it must define complex business logic including heterogeneous data formats, business rules, and business processes. WSCL restricts this by only allowing the definition of one group of Input & Output documents in an interaction -- not allowing processing of heterogeneous data. In addition, the standard definition of transition defines the condition for source interactions through a data format, and it does not consider heterogeneous business rules in the condition. In this work, we extend WSCL to satisfy the needs above. The SaaS application provider can use the extended WSCL (ex_WSCL) to handle the business model from his view. We also build a modeling tool as an interface to simplify the modeling process. The business model is meant to be implemented to run in the server of the SaaS application provider, and different tenant models form a subset of the business model. To assist the tenant, we build an algorithm to create a tenant model from the business model automatically in the modeling tool.

II RELATED WORKS

Personalization is proposed in the search engine firstly, and used to provide different services to different users for their different requirement [11]. In SaaS application, one instance is used by different tenants having different personalized needs. Consequently, the personalization also needs to talk in this area.

Sun[12] proposes the personalization in SaaS needs to consider the aspects like different organization structures, workflows, process data, user interfaces, reports and business rules. But he only proposes the problem itself, not provide the solution to how to do with the personalization. Zhang[13] proposes a policy-based approach for SaaS customization, this method provides good idea to solve the problem in personalization, but not provide a uniform description profile for SaaS, so it is not convenient to use the method widely.

For sake of the space, in this paper, we consider the personalization in the aspects of heterogeneous data, process rules, and business rules. Firstly, we use extended WSCL to describe SaaS application, and list the extension content for WSCL. Secondly, we take the policy-based approach for modeling process for personalization similarly. Thirdly, as the description profile for SaaS used in our paper is a conversation protocol, we provide the modeling tool and algorithm to help users to construct their SaaS application.

In the following section we present a SaaS application scenario. Section 4 illuminates the restrictions of standard WSCL for the aforementioned SaaS application scenario, and presents a SaaS Service model based on ex_WSCL. In section 5 we describe how to model a SaaS Service in the application, and present all of the personalized modeling processes from building a business model to obtaining a tenant model. Section 6 contains an explanation of the modeling tool for a SaaS Service, and presents an algorithm for producing a tenant model from a business model according to a personalized policy. Finally section 6 presents ongoing work and conclusions.

III SAAS APPLICATION SCENARIO

In the main business flow of a Customer Relationship Management System (CRM), if a salesman finds a cooperative customer's basic information, he could add this customer as a "lead." If salesman finds the customer satisfying certain rules, he could change the customer's role from "lead" to "account."

Alternatively, the salesman could collect the information of the cooperative customer, and add him as an "account" directly. After the customer becomes an "account," the salesman can forward sales opportunities to him. The salesman would promote customer interest in a deal. If the salesman fails, the whole flow ends; if the salesman succeeds, he converts the opportunity into a quote. The salesman submits the quote to the customer, and awaits the customer's reply. Then there are three possible situations. If the customer accepts the quote, the salesman sends the order to him, and the flow continues. If the customer would like to modify the quote, the customer indicates as such, and then the salesman quotes again. In the third situation, the customer rejects the quote, the salesman closes the quote and ends the whole flow. If the flow was successful, then following the signing of a contract and order generation, the salesman sends a message to other department s to arrange shipping, and to wait for the message from the financial department receiving the contract amount. When both the shipping and confirmation are complete, the financial department is directed to send the invoice to customer, and the flow ends.

When we apply SaaS in CRM applications, different enterprises may have different needs facing the management of their customer relationships. For example, Tenant A may only wish to obtain the minimal function set from the provider – containing only the management of "lead," "account," and "order." Conversely, Tenant B would like to utilize the entire flow above, but with the additional action "MoneyConfirm" to be included before shipping. Tenant C perhaps does not want his flow to contain the step of "quote." According to personalized needs from different tenants, we model their flows separately by using the UML Activity Diagrams in Figure 1.

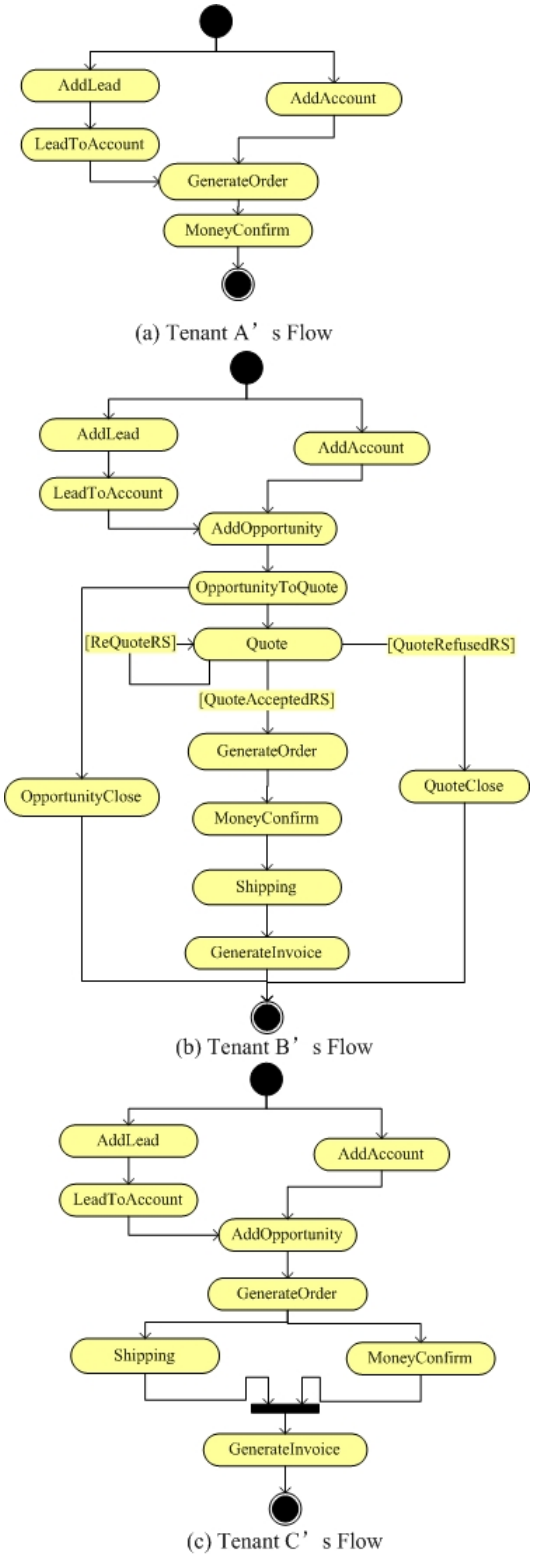


Figure 1. Tenants' flow

The SaaS application requires a Web-based software application deployed and operated as a hosted service over the Internet and accessed by users. Multi-tenancy SaaS

applications are typically installed in managed Internet data centers with remote management accessibility. In SaaS applications, there are four levels of SaaS maturity models. The level four model has a load-balanced form of identical instances with configurable metadata for its tenants. If our multi-tenancy SaaS application of CRM uses the fourth level of SaaS maturity model, we can build only one flow from the provider view, and it should satisfy all of the personalized needs from Tenants A, B, and C. We present this flow in Figure 2 with a UML Activity Diagram.

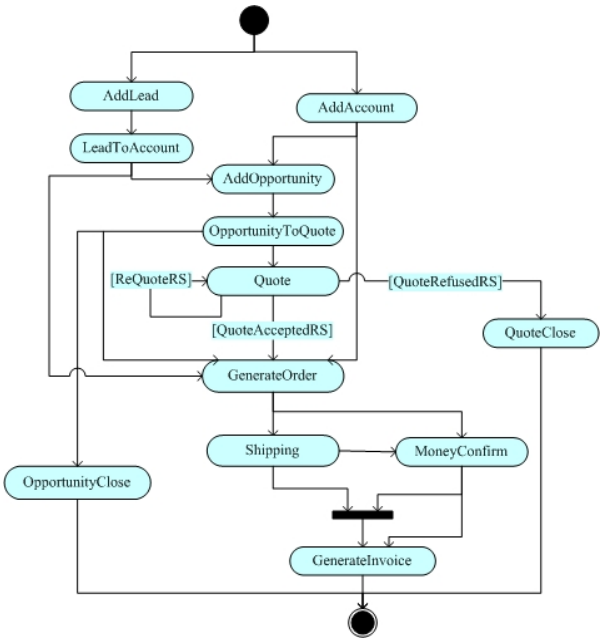


Figure 2. Provider's Flow

SaaS is a Web-based software application, while Web service technologies are implemented based on a set of XML standards, and can support interoperable application-to-application interactions over the Web. Consequently, applying Web service technologies in SaaS applications is becoming more popular. We can choose BPEL or WSCL for implementation of complex business logic models in SaaS applications of CRM based on Web service technologies. BPEL is a language to describe orchestration in service composition [9], and it only represents control from one party's perspective. When we describe the business logic of SaaS by BPEL, we could build it just from provider's view, as shown in Figure 2. However, each tenant is exposed to the complex business logic, and has difficulty separating his from others. There are also business security problems in this model, as one can easily see others' business rules. When we realize the logic, we must add numerous conditions to determine which type of tenant goes to which branch. For example, after action AddAccount, we could go to action AddOpportunity or GenerateOrder. The implemented process needs to add a condition to determine which type of tenant uses the application, then decides which action to go to next. If he is Tenant A, he continues to action GenerateOrder; if he is Tenant B or Tenant C, he continues to action AddOpportunity. Similar condition determination is very common in SaaS applications, so the implemented process complexity would increase significantly with many kinds of tenants.

WSCL is a language to describe choreography in service composition [9], and as shown in Figure 3 it provides a conversation model to allow each involved party to describe its part in the interaction. We build the entire complex business model first, then model each tenant according to his personalized needs, so each tenant only sees his own business logic. In this way, he easily views his business logic and avoids it being exposed to others. Business models and tenants' models based on WSCL run corresponding operations. When we implement the process, each tenant model defines the actions he would use. The business model can only run the corresponding operations, so we do not need to add determination conditions in the business model. In Tenant A's flow, he undertakes the action of GenerateOrder after AddOpportunity, and the implemented business flow from the provider would take the corresponding actions. In any such situation, the implemented business flow takes the actions corresponding to the tenant's model. The implemented business model handles asynchronous messaging in a conversational manner with many tenant models, thereby reducing running cost and making upgrading and maintenance simple. WSCL is therefore ideally suited to describing the SaaS Service model, but the existing WSCL framework restricts the expression of certain types of business processes, data formats, and business rules. To address this limitation, we extend the definitions of interface, transition, and business rules in WSCL to satisfy the needs of the SaaS application. We also build a modeling tool as an interface to ease the modeling process. For convenience to tenants, we build an algorithm to help them obtain tenant models from the business model automatically in the modeling tool.

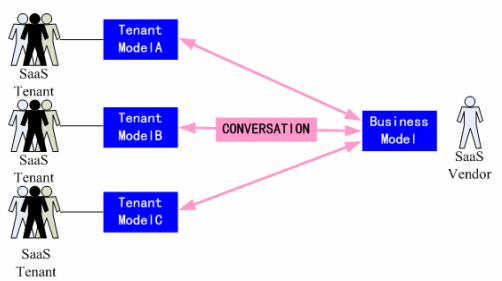


Figure 3. SaaS application Using Conversation Model

IV SAAS SERVICE MODEL BASED ON EX_WSCL

WSCL allows each involved participant in the conversation to express his business logic himself, so it is suitable for modeling a SaaS Service. However, WSCL has restrictions about expressing heterogeneous data formats, business rules, and business processes in one flow. In the definition of WSCL, conversations contain interaction and transition. Interaction models the actions of the conversation as document exchanges between two participants. One interaction maps to one document exchange group, and supports five types, which are Receive, Send, ReceiveSend, SendReceive, and Empty. As the SaaS application is required to run one instance to fit all of its tenants, a one-to-one mapping in interaction can not satisfy the needs of heterogeneous data formats. Transition specifies the ordering relationships between interactions. A transition specifies a source interaction, a destination interaction, and, optionally, a document type of the source interaction called SourceInteractionCondition as an additional condition for the transition. Only document type restriction in transition is not sufficient, so it must add flexible definition of business rules. For example, in the CRM when the Opportunity turns to a Quote, the salesman may provide different prices to different customers. Different enterprises may implement different rules for discounts; some may consider the purchase amount of goods, some may consider the customer's rating. So we need to add flexible business rules for discount policies in the transition from action AddOpportunity to OpportunityToQuote. In addition, one transition only has one source interaction and one destination interaction, and thus can not be used to model concurrent structure in the flow. For example, for the flow in Figure 2, before the action of GenerateInvoice begins, both of the actions Shipping and MoneyConfirm should end. The transition between them requires two source interactions. We therefore extend WSCL in three directions as explained below.

A. *Interface extension*

One Interaction does not map to one group of information exchange, but to several groups. This means that one interaction could bind with several groups of input and output. For example, the interaction of AddAccount in Figure 2 could map two groups of input-output after extension to satisfy different tenants' personalized needs for data format.

```
<Interaction id=" AddAccount" Interaction Type="ReceiveSend">
    <Group id="Account1">
      <InboundXMLDocument hrefSchema="http://uddi.
    neu.edu.cn/CRM/Account1.xsd" id="AccountRQ1"/>
      <OutboundXMLDocument hrefSchema="http://uddi.
    neu.edu.cn/ ValidApplyRS.xsd" id="ValidAccountRS" />
      <OutboundXMLDocument hrefSchema=" http://uddi.
    neu.edu.cn/ InvalidApplyRS.xsd" id="InvalidAccountRS"/>
    </Group>
    <Group id=" Account 2">
      <InboundXMLDocument hrefSchema="http://uddi.
     neu.edu.cn/CRM/Account2.xsd" id="AccountRQ1"/>
      <OutboundXMLDocument hrefSchema="http://uddi.
     neu.edu.cn/ ValidApplyRS.xsd" id="ValidApplyRS" />
      <OutboundXMLDocument hrefSchema=" http://uddi.
     neu.edu.cn/ InvalidApplyRS.xsd" id="InvalidApplyRS"/>
    </Group>
</Interaction>
```

B. *Transition extension*

One transition only contains one SourceInteration and one DestinationInteraction, but would not be suitable for defining concurrent structured transitions. In concurrent structured transitions, the exchange document of several SourceInteractions must all satisfy the constraints and the transition triggers for the DesitionationInteraction to be executed. We therefore extend the definition of transition to allow it to have several SourceInterations and only one DestinationInteraction. We also modify the transition condition SourceInteraction, and allow it not only to contain elements of the exchange document but also their compound expressions.

```
<Transition>
    <SourceInteraction href="Shipping">
    <SourceInteraction href="MoneyConfirm">
    <DestinationInteraction href="GernerateInvoice">
    <SourceInteractionCondition href="ShippingRS" and "
     MoneyConfirmRS"/>
</Transition>
```

C. *Business rules extension*

When triggering a transition in a business process, there are business rules for its precondition except the document type restriction in source interactions. We add business rules in the definition of transition and require them to be part of compound logic expressions to constrain attributes in the interaction's I/O. For example, we can describe business rules in the transition from interaction AddOpportunity to OpportunityToQuote in Figure 2.

```
<Transition>
    <SourceInteraction href="AddOpportunity">
    <DestinationInteraction href="OpportunityToQuote">
    <SourceInteractionCondition href="OpportunityRS"/>
    <BusinessRules>
      <Rule>if Account.Rating=3 discount=0.7;
            else if Account.Rating=2, discount=0.9
      </Rule>
      <Rule>if Quote.Amount>100,000 discount=0.85
            else if Quote.Amount>10,000 discount=0.8
      </Rule>
    </BusinessRules>
</Transition>
```

WSCL is suitable for modeling a SaaS Service, but its existing definition restricts the expression of heterogeneous data formats, business rules, and business processes, and therefore can not satisfy multiple tenants in SaaS application,s. We therefore extend WSCL with modified interface, transition, and business rules. We can model the complex business logic in SaaS with extended WSCL. In the next section, we describe the modeling process in a SaaS service and how to get the Tenant Model based on Business Model.

V SAAS SERVICE MODELING

A. *Personalized Modeling Process*

In this paper, SaaS service model based on extended WSCL uses conversation as basic element for describing its business logic. For one interaction in the conversation, Tenant Model sends a message to Business Model and waits for receipt of response, while Business Model receives the message from Tenant Model and sends its response to it. Both parties involved in the conversation have the same interaction and interact with each other by running corresponding operations. If we consider only interactions in the conversation, and ignore both parties having the different operations in the same interaction, we find that each tenant model is a subset of the business model, and the business model is the union set of all the tenant models. When constructing a SaaS application based on Web Service, the provider could construct a main process firstly, and then describe it as Business Model by extended WSCL. When a tenant wishes to subscribe to the SaaS application, he could choose the interactions and a sequence of them included in business model. According to his choice, the provider separates their jointly owned interactions from the business model, inverts the operations in them, and produces a tenant model. The tenant gets his model from his view, and does not need to know all of the complex business logic in the business model. If he wants more functionality or a different business flow, he could send his personalized requirement to the provider. The provider may modify his business model by adding new interactions or transitions or new interface definitions, and publish a new model.

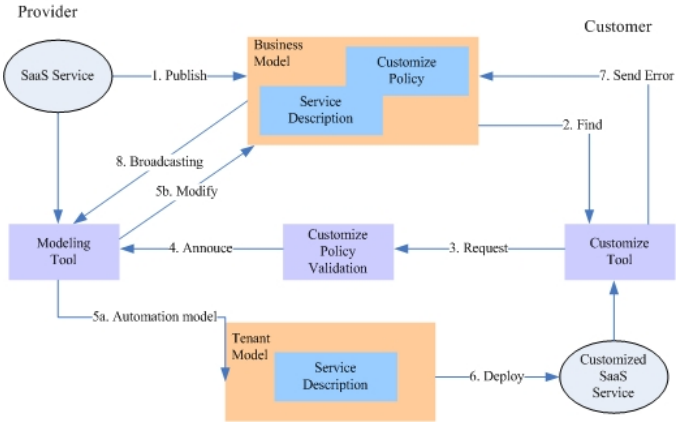


Figure 4. Personalized Modeling Process for SaaS Service

The modeling process includes constructing the business model and producing a personalized tenant model as shown in Figure 4. We divide the process into eight steps and describe it as follows:

Step 1. The SaaS service provider designs a business model, validates its correctness by formal validation, and publishes the business model on the Web;

Step 2. The tenant discovers the SaaS service, sends his personalized policy including choice of interactions and sequences from the business model;

Step 3. The policy is sent to policy validation to determine whether it exists or not;

Step 4. Policy validation sends the result of validation to provider;

Step 5. If the policy exists, the modeling tool produces the tenant model according to policies. If the policy does not exist, the provider modifies his business model and validates its correctness. The provider then publishes the new tenant model and updates others by broadcasting;

Step 6. The tenant gets his new model, deploys it and runs it;

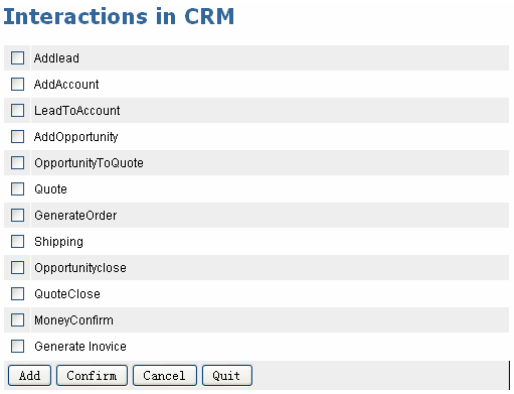
Step 7. If the tenant finds errors during operation, he informs the provider;

Step 8. The provider modifies his business model according to error information and broadcasts the new model to all tenants.

B. Personalized Policy

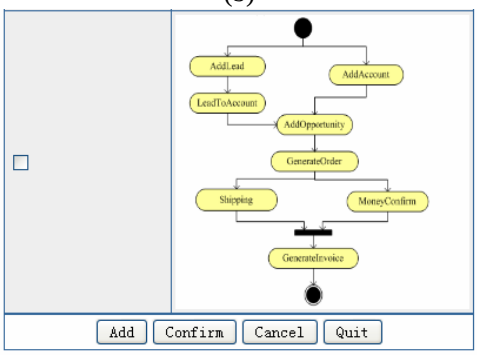
Personalized Policy is very important in our work, and tenants can use it to build their flows based on provider's flow. We provide four kinds of basic operations in the policy definition: (1) choosing interactions, process, I/O in the interaction, and business rules; (2) adding interactions, process, I/O in the interaction, and business rules; (3) deleting interactions, process, I/O in the interaction, and business rules; (4) modifying interactions, process, I/O in the interaction, and business rules.

To simplify the definition of the tenants' policies, we provide an interface as a constructor to help tenants in the process of defining their policies. Shown in Figure 5, the interface firstly defines the interactions in CRM, and the system provides the existing process flows. The tenant would also define the I/O in the interaction and the business rules in the transition. All of the operations by tenants would generate a profile including his policies and pass it to be verified by the policy validation tool.



(a)

(b)



(c)

(d)

Figure 5. Interface for Defining Personalized Policy

C. Modeling Tools

For convenience to both the provider and the tenants, we provide a modeling tool. We can use it to create an effective state diagram and to document WSCL for business service. The tool implements two types of functional modules, which are SaaS Service state diagram modeling and WSCL document conversion with state diagrams. The first functional model is implemented with graphical editor framework (GEF) and rich client platform (RCP). The second functional model utilizes reverse engineering to transform WSCL into a state diagram and vice versa. The interface for the modeling tool is shown in Figure 6.

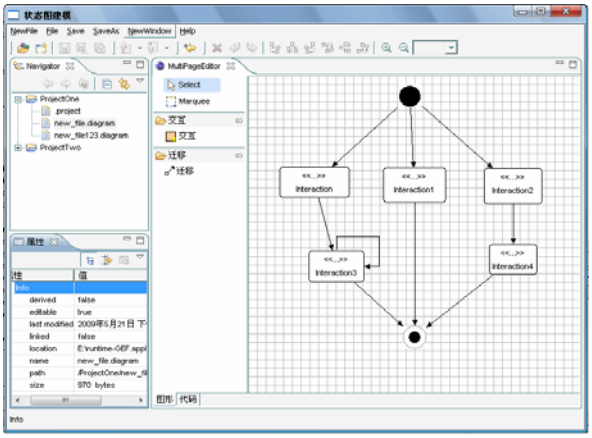


Figure 6. Modeling Tool for SaaS Service

In addition, the tool provides for obtaining the tenant model from the service model according to the tenant's personalized policy automatically. The algorithm to implement the automation process is as follows. Firstly, we transform the WSCL document of the service model to a graph, as described by the work of Daniela et al[10]. Secondly, as we focus on different processes for customization in this paper, we describe how to implement to choose one sub-process from several ones and delete the unselected branches in the process.

```
Algorithm1 Get sub-graph according to the path user chooses.
  CreateSubGraph(Graph G, Vector Paths, int i) {
     Path p=DiffPath(Paths, i);
     if(p!=null){
      for(i=1,s=p.firstarc();i<p.getarcnum();i++){
            G.DeEdge(s);//Delete an arc s from Graph G
            s=p.getnextarc();}//
     }
     CheckGraph(G);
  }
  Algorithm2 Find the Difference Set from user's chosen path with other paths.
  DiffPath(Vector Paths, int i) {
     Path p1=Paths.getUnion(i);//Get the union set of other paths except user's chosen path
     Path p2=Diff(p1,Paths.get(i));//Get the difference set between p1 and user's chosen path
     Return p2;
  }
```

VI CONCLUSIONS

This paper proposes a modeling method based on Web services for SaaS applications with multi-tenancy support. We use a conversation manner to describe the business logic in the application. Considering the existing WSCL lack of support for heterogeneous data formats, business rules, and business processes, we extend it through interface, interaction and transition. The model used in this paper is composed of a business model and a tenant model, and the two parts involved in the conversation have the same interaction but individualized operations. Through guidelines in the conversation manner, we also present the procedure for obtaining the tenant model according to business model automatically. Finally we put forward the whole modeling process, introducing the modeling tool and algorithms to add convenience. We have constructed a demonstration version to run the SaaS application in this way. Adding running information in WSCL is the goal of our future work.

REFERENCES

[1] Iod, Sotware as a Service, Kogan Page Ltd, 2002.

[2] OASIS, Service Oriented Architecture (SOA), http://xml.coverpages.org/soa.html.

[3] Patrick C. K. Hung, Wendy Hui ."Software as a Service (SaaS): Security Strategy, Risk Management, Static Analysis and Assessment Tool", SCC2008 Tutorial 4.

[4] Sun Wei, Kuo Zhang, Shyh-Kwei Chen, etc. "Software as a Service: An Integration Perspective", ICSOC 2007, pp 558-569.

[5] Thomas Kwok, Thao Nguyen and Linh Lam.A. "Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application", 2008 IEEE International Conference on Services Computing, pp 179-186.

[6] WSDL, http://www.w3.org/TR/wsdl

[7] BPEL,http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[8] WSCL, http://www.w3.org/TR/wscl10/

[9] Chris Peltz, Web Services Orchestration and Choreography, Computer, 46-52

[10] Daniela G, Juan C C, Mokrane B.Behavioral matchmaking for service retrieval. ICWS '06. International Conference on Sept, 2006 ,145-152

[11] Pretschner, A. Ontology based personalized search [MS. Thesis]. Lawrence, KS: University of Kansas, 1999.

[12] Wei Sun, Xin Zhang, Chang Jie Guo, Pei Sun, Hui Su. Software as a Service: Configuration and Customization Perspectives, 2008 IEEE Congress on Services Part II, 18-24

[13] Kuo ZHANG, Xin ZHANG, Wei SUN, Haiqi LIANG, Ying HUANG, Liangzhao ZENG and Xuanzhe LIU, A Policy-Driven Approach for Software-as-Services Customization, The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)