A Cloud Workflow Modeling Framework Using Extended Proclets

Hua Huang[1,2], Rong Peng[1(✉)], Zaiwen Feng[1], and Min Zhang[2]

[1] State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China
{rongpeng, zwfeng}@whu.edu.cn,
[2] School of Information Engineering, Jingdezhen Ceramic Institute, Jingdezhen 333001, China
{jdz_hh, jdz_zm}@qq.com

Abstract. Most existing cloud workflow modeling approaches focus on how to support business process customization for multiple tenants. But there are many factors to be considered in actual cloud workflow modeling, such as how to model interaction between different tenants' business processes while protecting their privacy respectively, and how to facilitate the reuse of common process fragments for different stakeholder roles. To address these issues, this paper proposes a cloud workflow modeling framework using extended proclets. The framework provides a hierarchical management mechanism to isolate the private data of each tenant's business process and adopts two-level-channel transmission protocol to explicitly model interaction between different tenants' business processes. At last, a case study is carried out to illustrate its modeling capability and feasibility.

Keywords: Cloud workflow · Proclets · Interaction between multi-tenant business processes · Hierarchical management mechanism

1 Introduction

Cloud workflow is a workflow management system deployed in cloud computing environment. In cloud workflow, different tenants can design, configure and run their business processes with three-level isolation: data isolation, performance isolation and execution isolation [1]. However, interaction between business process instances of different tenants (e.g. cross-enterprise collaborative business processes) in a cloud workflow system is necessary to be considered. Additionally some common process fragments should be reused for different stakeholder roles. For instance, there exists at least three types of enterprise stakeholder roles (i.e. three types of tenants) in a Cloud Distribution Resource Planning System (denoted as CDRP), i.e. supplier (ceramic product supplier), online distributor (selling suppliers' ceramic products only in online store), physical distributor (selling suppliers' ceramic products in both online store and physical store). Although they share some common processes fragments, e.g., registering and creating online store process, each tenant has its personalized process, e.g., the process of building distribution channel for suppliers and the process of building

J. Bae et al. (Eds.): AP-BPM 2015, LNBIP 219, pp. 19–34, 2015.
DOI: 10.1007/978-3-319-19509-4_2

procurement channel for distributors. In order to complete the product distribution business, interaction between their personalized processes should be achieved. In general, interaction between cross-enterprise collaborative process instances is hidden in customized applications through hard-coded business process definition languages, which contributes to the difficulties for updating and maintaining the corresponding system. Therefore, it is necessary to explicitly model interaction between different stakeholder roles in CDRP. Unfortunately, another challenge emerges when modeling interaction between multi-tenant business processes. That is how to isolate and protect each tenant's business processes related private data [2].

To solve these problems, we adopt hierarchical management ideology to extend the proclets framework [5, 6] for cloud workflow modeling. In summary, we make the following contributions in this paper:

- Propose a novel cloud workflow modeling framework based on extended proclets, which adopts two-level-channel transmission protocol to explicitly model interaction between different tenants' business processes. Meantime, it provides a hierarchical management mechanism to protect their private data.
- A case study is conducted within a CDRP System to demonstrate the feasibility of the proposed approach.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 proposes a cloud workflow modeling framework using extended proclets. In Sect. 4, we illustrate our approach by modeling three types of tenants' (suppliers, online distributors, physical distributors) business processes in CDRP. Section 5 concludes this paper.

2 Related Work

Recently, cloud-based business process modeling approaches are emerging. For instance, Mietzner [7] presents a customizable process model by individualizing a given process template based on tenant's requirements, and then converting it into a BPEL process model. In [8], BPEL is extended into VxBPEL, which enables business processes to be individualized based on variable points. And Ralph also adopts variable points to customize BPEL process model [7, 9, 10]. However, most cloud-based business modeling methods focus on multi-tenant business process customization without considering interactions between different tenant business processes.

Meanwhile, the tools and languages for modeling cloud-based business processes are also to extend the general modeling notation, e.g. Petri net, BPMN, EPC, UML AD, BPEL, etc. These approaches focus on isolated case-based processes without considering the interaction between multiple business processes. In recent years, the object-oriented and artifact-centric modeling approaches provide a solution combining process-centric and data-centric process modeling approaches [3, 4]. The object-oriented modeling approaches adopt objects to denote the information entities that capture business goals [3]. Compared with traditional business process modeling approaches, they focus on modeling business contexture and behavior, rather than activity sequences. With respect to the artifact-centric modeling approaches, the business

process is defined by four components: business artifact, life-cycle, business rules and services [4]. These approaches use business artifacts to combine data-flow and control-flow in a holistic manner as the basic building blocks. Thus, it is easier to implement the migration of business processes. Although the advantages of the object-oriented and artifact-centric modeling approaches have been evaluated both in academic research and industrial applications, they have not yet been extended to cloud workflow modeling. In addition, there is another business process modeling approach based on proclets framework [5, 6]. A proclet is light-weight workflow process, which decomposes a big and complex process into several interacting sub-processes. The proclets framework addresses the issues of communication and collaboration of internal sub-processes. Moreover, it is possible to model complex workflows in a more natural manner by proclets. It means that the proclets framework is helpful for solving the problems mentioned in Sect. 1, e.g., interaction between multi-tenant business processes. In order to use proclets to model cloud workflow, it is necessary to extend proclets framework based on hierarchical management ideology.

3 Cloud Workflow Modeling Framework Using Extended Proclets

The proclets framework is composed of four main components: proclet, channel, naming service, and actor. The original framework of proclets [6] is shown in Fig. 1.

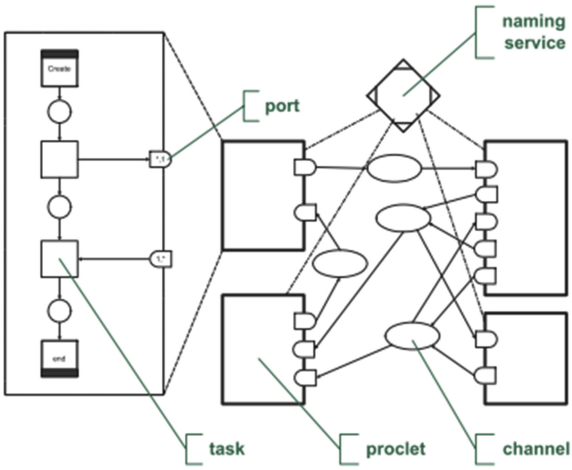

Fig. 1. The original framework of proclets

In Fig. 1, a proclet can be seen as a light-weight workflow or a process object with life-cycle. It can interact with other proclets via communication channels, which are the medium to transmit messages from one proclet to another. A task (or transition) can send or receive messages via ports. In the original framework, all proclet instances are

monitored and managed by a naming service. The detail of each component can be found in [6]. To satisfy new requirements for cloud workflow modeling, we adopt hierarchical management method to extend channel, naming service, knowledge base and proclet class except for actor, which is packaged into proclet class as process related data (namely role).

3.1 Extending Channel

Channels are used to link multiple proclets and transmit messages between them. To achieve transmitting messages between different tenant-level proclets (Definition in Sect. 3.4) in cloud workflow, we extend channel into two categories: tenant-level channel and tenant-inner-level channel. Tenant-level channel (depicted as a double-solid-line ellipse) is used to link tenant-level proclets for transmitting messages between them. Tenant-inner-level channel is similar to the channel in Fig. 1, but it includes three types: tenant-inner-level input channel (denoted as *Inner Channel for Input*), tenant-inner-level output channel (denoted as *Inner Channel for Output*) and tenant-inner-level communication channel (denoted as *Inner Channel*). Tenant-inner-level input channel (depicted as a single point-line ellipse) is used to link tenant-level proclets and tenant-inner-level proclets (Definition in Sect. 3.4) for transmitting messages from tenant-level proclets to tenant-inner-level proclets, while tenant-inner-level output channel (depicted as a single-dotted-line ellipse) is used to link tenant-inner-level proclets and tenant-level proclets for transmitting messages from tenant-inner-level proclets to tenant-level proclets. Tenant-inner-level communication channel (depicted as a single-solid-line ellipse) is used to link tenant-inner-level proclets for transmit messages between them.

3.2 Extending Naming Service

In Fig. 1, all interaction between proclets is based on proclet identifiers (*proc_id*) and class identifiers (*class_id*). These identifiers provide the handles to route communication information. In general, the sending proclet does not know the p*roc_id* of all receiving proclets. Thus, the original proclets framework [6] adopts naming service to keep track of all proclets. The naming service provides four basic functions: registering, updating, querying, and unregistering proclets' related information, such as *proc_id*, *class name, creator, creating time, owner* and *key attributes*. In the extended proclets framework, the naming services can be grouped into two categories: global naming service (depicted as a double-line diamond) and inner naming service (depicted as a single-line diamond). In a cloud workflow system, there is only one global naming service, which is used to register, query, update, and unregister all tenant-level proclets. The inner naming service is defined in a tenant-level proclet. For any tenant, there exists only one tenant-level proclet, thus there is only one inner naming service to manage its tenant-inner-level proclets.

3.3 Extending Knowledge Base

A proclet uses knowledge in its knowledge base to make routing decisions. In order to support interactive communication between tenant-level proclets, we extend knowledge base into two categories: tenant-level knowledge base and tenant-inner-level knowledge base. Tenant-level knowledge base is defined in a tenant-level proclet. It mainly consists of a communication log table, which is used to keep all records of communication information related to the tenant-level proclet, including communication time, channel name, sender, *proc_id* of sender's tenant-level proclet, receivers, *proc_ids* of receivers' tenant-level proclets, action, scope (Public or Private), etc. While tenant-inner-level knowledge base is defined in a tenant-inner-level proclet. It also mainly contains a communication log table, in which the stored information is similar to the tenant-level knowledge base except for *proc_id* of sender's tenant-level proclet and *proc_ids* of receivers' tenant-level proclets.

3.4 Extending Proclet Class

In Fig. 1, the life-cycle of a proclet instance and all its ports is described via a proclet class (denoted as *PC*), which can be compared to an ordinary workflow process definition or workflow type. In the original proclets framework, the proclet class describes the execution order of all tasks using a graphical language based on Petri nets. In this paper, we adopt the artifact-centric modeling approach to package the process related data (e.g., the executor and service resources needed for implementing tasks of business processes, etc.) and the life-cycle of a proclet instance into a proclet class. The activity sequences involved in the life-cycle are represented by Petri nets. To support hierarchical management, proclet class is extended into two categories: tenant-level proclet (denoted as *Tenant_PC*) and tenant-inner-level proclet.

Tenant_PC is a virtual process object created for each tenant. It is used to monitor and manage all tenant-inner-level proclets. A tenant-level proclet class has a unique identity (*class_id*). Meanwhile, each tenant corresponds to only one tenant-level proclet instance with unique identity (*proc_id*). The *proc_id* is consistent with tenant account (denoted as *tenant ID*). A tenant-level proclet class is composed of four parts: tenant-level knowledge base, tenant process related data set, life-cycle, and inner naming service. The tenant process related data set contains a set of tenant related attributes (*tenant ID, tenant name, tenant type, tenant profile*), a set of tenant-inner personnel roles and a set of service resources for implementing tenant business processes. The class structure of *Tenant_PC* is shown in Fig. 2.

In Fig. 2, the life-cycle of *Tenant_PC* is composed of four transitions (*Create, Input Processing, Output Processing* and *End*), three places (*P1,P2,P3*) and four ports (*Outer Input, Outer Output, Inner Output, Inner Input*). The transition *Create* is activated to generate its instance when a tenant is registering. The transition *End* is executed to destroy its instance when a tenant is unregistering. The transition *Input Processing* is linked by two ports: *Outer Input* and *Inner Output*. The port *Outer Input* is used for receiving communication messages from other *Tenant_PC* instances through the tenant-level channel, then the communication messages are submitted to the transition

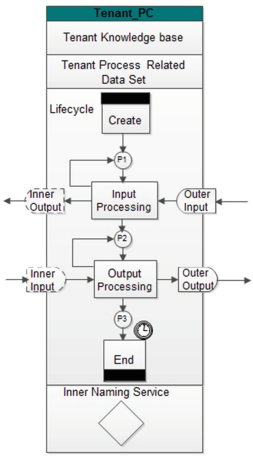

Fig. 2. The class structure of *Tenant_PC*

Input Processing for handling operation, e.g., checking message, decrypting ciphertex, sending classification and saving log (storing the routing information into the log table of tenant-level knowledge base), etc. The port *Inner Output* is used to send communicate messages handled by the transition *Input Processing* to the receiver (a tenant-inner-level proclet instance) through the tenant-inner-level input channel. The transition *Output Processing* is also linked by two ports: *Inner Input* and *Outer Output*. The port *Inner Input* is used for receiving communication messages from tenant-inner-level proclet instances through the tenant-inner-level output channel, then the communication messages are submitted to the transition *Output Processing* for handling operation, e.g., coding message, encrypting related private data, sending classification and saving log, etc. The port *Outer Output* is used to send communicate messages handled by the transition *Output Processing* to other *Tenant_PC* instances through the tenant-level channel.

Tenant-inner-level proclet is an actual process object, which is used to encapsulate each tenant's business process and its related business data with a certain size granularity. It is divided into two categories: shared tenant-inner-level proclet (denoted as *Shared_PC*) and general tenant-inner-level proclet (denoted as *Inner_PC*).

Shared_PC is used to encapsulate common process objects and related metadata for the same type tenants. According to whether *Shared_PC* can be customized, it can be extended into two categories: fixed shared tenant-inner-level proclet (denoted as *Fixed_Shared_PC*) and customizable shared tenant-inner-level proclet (denoted as *Customizable_Shared_PC*). *Fixed_Shared_PCs*, which are common business process objects pre-designed and stored in process repository, can be reused without any modification. Whereas they cannot be customized. As shown in Fig. 3, a *Fixed_Shared_PC* class is composed of three main components: tenant-inner-level knowledge base, shared process related data set and life-cycle. The life-cycle of a *Fixed_Shared_PC* class is mainly composed of many transitions and places. A transition is linked by ports (Input Port or Output Port). The Input Port is used for receiving communication messages transmitted through tenant-inner-level channels,

and the Output Port is used for sending communication messages from the linked transition. The shared process related data set mainly contains a set of public business attributes (e.g., *business ID, business name, business type, business description*, etc.) and a set of relations between transitions and resources (including roles and services used for executing business activities), etc. *Customizable_Shared_PCs* are also common business process objects pre-designed in cloud workflow system or defined by tenants. They can be reused by making modification (e.g., adding or deleting transitions, setting the input and output ports, modifying business metadata, etc.). As opposed to *Fixed_Shared_PCs*, they can be customized according to tenants' requirements. As shown in Fig. 3, a *Customizable_Shared_PC* class is composed of three components: tenant-inner-level knowledge base, customizable process related data set and life-cycle. Although the life-cycle and customizable process related data set of *Customizable_Shared_PC* is similar to the definition of *Fixed_Shared_PC*, the former can be customized and the latter cannot.

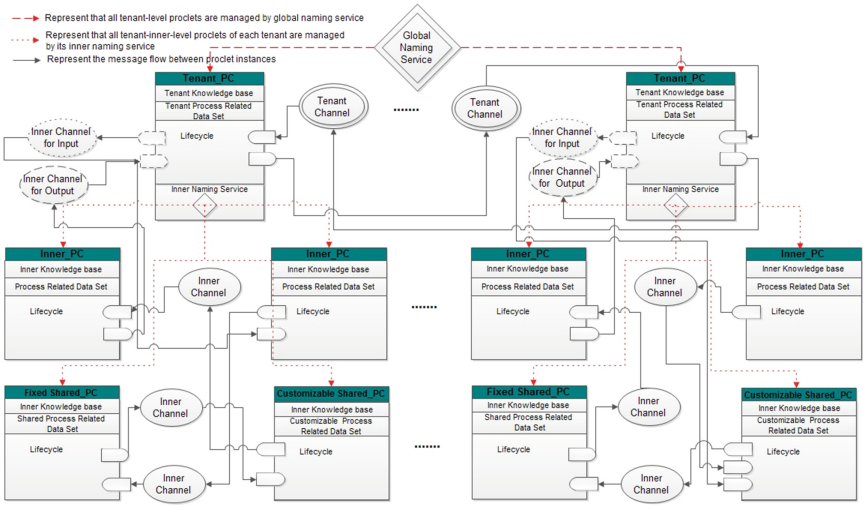

Fig. 3. The cloud workflow modeling framework based on extended proclets

Inner_PC is defined by tenants. During the modeling process, according to multiple business goals or business artifacts, a big and complex business process can be decomposed into multiple sub-processes that can be easily modeled and implemented. These sub-processes can be modeled as *Inner_PCs*. As shown in Fig. 3, a *Inner_PC* class is composed of three components: tenant-inner-level knowledge base, process related data set and life-cycle. The life-cycle and process related data set of *Inner_PC* is similar to the definition of *Customizable_Shared_PC*.

According to the above description for extending proclets, the cloud workflow modeling framework based on extended proclets is proposed as shown in Fig. 3.

In Fig. 3, there exists only one global naming service, which is used for monitoring and managing all tenant-level proclet instances. For each tenant, all tenant-inner-level proclet instances belonging to it are monitored and managed by its inner naming service.

Interaction between tenant-inner-level proclet instances belonging to the same tenant-level proclet instance is easily achieved by tenant-inner-level communication channels. However, it is difficult to achieve interaction between tenant-inner-level proclet instances belonging to different tenant-level proclet instances. For example, in Fig. 3, a tenant-inner-level proclet instance (assumed as *PC1*) owned by a tenant-level proclet instance (assumed as *Tenant_PC1*) sends a communication message (assumed as *MSG*) to another tenant-inner-level proclet instance (assumed as *PC2*) owned by another tenant-level proclet instance (assumed as *Tenant_PC2*). Firstly, *MSG* is created by a transition (assumed as *T1*) in *PC1*, then *MSG* is sent by the Output Port linked to *T1*, and transmitted to the Inner Input Port of the *Tenant_PC1* through the tenant-inner-level output channel. *MSG* is submitted to the transition *Output Processing* of *Tenant_PC1* for security handling (e.g., security coding, data encryption, etc.). After having been handled, *MSG* is sent by the Outer Output Port of *Tenant_PC1*, and transmitted to the Outer Output Port of *Tenant_PC2* through the tenant-level channel. Then, *MSG* is submitted to the transition *Input Processing* of *Tenant_PC2* for security handling (e.g., security authentication, cipher text decryption, etc.). After having been handled, *MSG* is sent by the Inner Output Port of *Tenant_PC2*, and transmitted to the Output Port linked to the receiver transition (assumed as *T2*) in *PC2* through the tenant-inner-level input channel. Finally, *T2* receives *MSG* successfully and be fired. It means that one interaction between *PC1* and *PC2* is completed without disclosing their private data.

In order to better understand the proposed cloud workflow modeling framework based on extended proclets, a case using this framework to model tenants' business process in CDRP is presented as below.

4 Case Study

The CDRP mentioned in Sect. 1 is designed to enhance the communication and interaction between supplier, online distributor, and physical distributor and to implement online ceramic product sale business, e.g., publishing ceramic product information, managing order, etc. In order to apply the services in CDRP, all tenants need to register and create online stores. For suppliers, when having registered and created their online stores, they are able to build distribution channels and online supply ceramic products. For online distributors or physical distributors, they also need to build purchasing channels to online purchase and sell ceramic products after having registered and created their online stores. It is clear that there exists interactions between the distribution channel processes built by suppliers and the purchasing channel processes built by distributors. In addition, there exists interactions between suppliers' online supplying ceramic products processes and distributors' online purchasing and selling ceramic products processes. Meanwhile, for online distributors and physical distributors, there are some differences in purchasing and selling ceramic products process. The online distributors make online selling ceramic products without stock, so they can directly download ceramic products authorized by their suppliers to their shops for sale. When a buyer creates an order in a online store, the corresponding purchasing order will be generated to the corresponding supplier. With respect to physical distributors, they can download ceramic products authorized by their suppliers

after having purchased them. Then, they can sell these ceramic products online or offline in their online or physical stores.

4.1 Description of Modeling Each Process Object

According to the aforementioned description in CDRP, three types of tenants (suppliers, online distributors and physical distributors) have their own business processes when they are using the services provided by CDRP. Among these processes, some sub-process or process fragments are identical or similar. To reuse them and simplify business process modeling for each tenant, tenant's business process are decomposed into multiple sub-processes based on sub-targets within a certain granularity; then, package each sub-process and related process data into a process object defined by a proclet. Thus, the supplier's business process can be decomposed into three sub-process objects: registering and creating online store, building distribution channels, and supplying ceramic products. The online distributor's business process can be decomposed into four sub-processes objects: registering and creating online store, building the purchasing channels, purchasing ceramic products and online selling ceramic products. It is the same as the physical distributor's business process except for online or offline selling ceramic products. Obviously, the common sub-process of registering and creating online store can be used by three types of tenants. Therefore, it is modeled as a *Fixed_Shared_PC* (denoted as *Create_Shop_Fixed_Shared_PC*). The sub-process of building the purchasing channels, in which there are some differences between online distributors and physical distributors, is modeled as a *Customizable_Shared_PC* (denoted as *Building_Procurement_Channel_Customizable_Shared_PC*). This is the same as the sub-process of purchasing ceramic products (denoted as *Product_Purchase Customizable_Shared_PC*). Other sub-processes will be modeled as *Inner_PCs*, i.e. *Building_Distribution_Channel_Inner_PC*, *Product_Supply_Inner_PC*, *Online_Sale_Inner_PC* and *Online_and_Offline_Sale_Inner_PC*. For better understanding how to define three types of *Inner_PC*s, we select three *Inner_PC*s: *Create_Shop_Fixed_Shared_PC*, *Building_Procurement_Channel_Customizable_Shared_PC* and *Online_Sale Inner_PC* as examples to describe their model structure.

(1) ***Create_Shop_Fixed_Shared_PC***

The sub-process object of registering and creating online store (denoted as *Create_Shop_Fixed_Shared_PC*) is pre-designed as a *Fixed_Shared_PC* in CDRP. It can be reused for three type of tenants when they are modeling their business processes. Meanwhile, it can be instantiated at runtime and monitored by the tenant-level proclet. Its model diagram is shown in Fig. 4.

In Fig. 4, the message places are marked up as P1 to P5. The transition *Create* is used to create proclet instance, and the transition *End* is used to destroy proclet instance. There are other transitions marked up as T1 to T4, the details of them are described as below.

T1: Register information.
T2: User login.
T3: Create online store.
T4: Decorate online store.

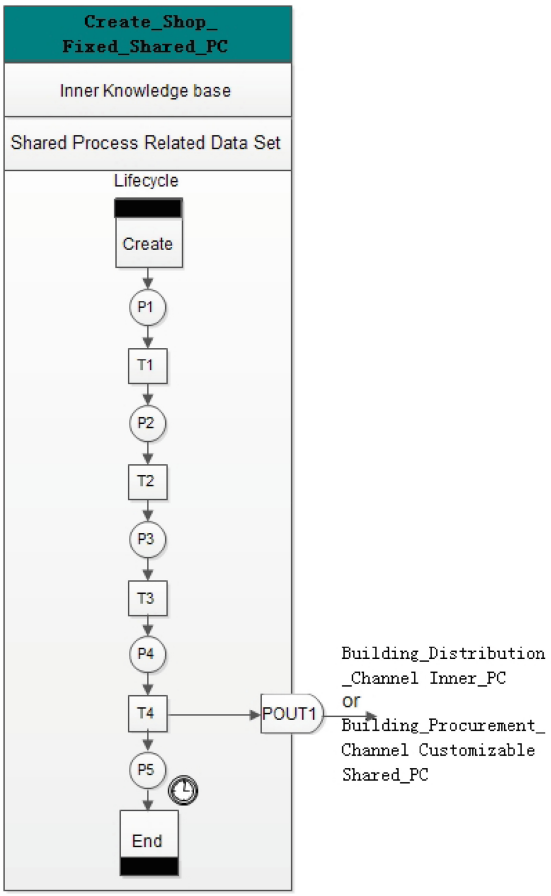

Fig. 4. Model diagram of *Create_Shop_Fixed_Shared_PC*

POUT1: An output port, one end of which is connected to T4, and the other end is pointed to *Building_Distribution_Channel_Inner_PC* or *Building_Procurement _Channel_Customizable_Shared_PC*. This means that after having decorated online store (or T4 is executed), for suppliers, T4 will send a communication message to activate *Building_Distribution_Channel_Inner_PC* though POUT1; but for distributors, T4 will send a communication message to activate *Building_Procurement_Channel_Customizable_Shared_PC*.

(2) *Building_Procurement_Channel_Customizable_Shared_PC*

The sub-process object of building procurement channels (denoted as *Building_Procurement_Channel_Customizable_Shared_PC*) is pre-desinged as a *Customizable_Shared_PC* in CDRP. It is used to apply for distribution eligibility and download products for distributors. For online distributors, they can directly download the

ceramic products authorized by their suppliers, yet physical distributors can download the ceramic products authorized by their suppliers only after having purchased them. Therefore, the communication message of finishing order is necessary to be considered. Its model diagram is shown in Fig. 5.

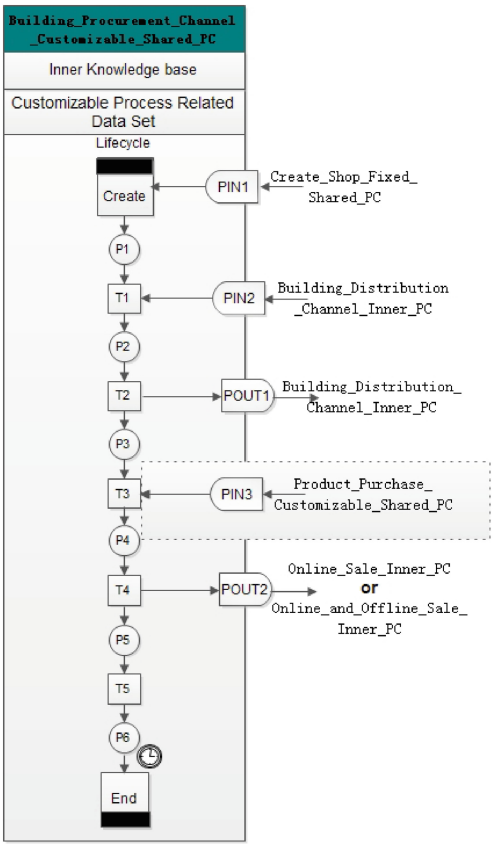

Fig. 5. Model diagram of *Building_Procurement_Channel_Customizable_Shared_PC*

In Fig. 5, the message places are marked up from P1 to P6. The transition *Create* is used to create the proclet instance, the transition *End* is used to destroy the proclet instance. There are other transitions marked up from T1 to T5. The details are described as below.

T1: Find suppliers.
T2: Apply for distribution eligibility.
T3: For online distributors, T3 represents downloading ceramic products authorized by their suppliers; for physical distributors, T3 represents downloading ceramic products purchased by them.
T4: Show ceramic products for online sale.
T5: Manage suppliers' information.

PIN1: An input port, one end of which is connected to the transition *Create*, and the other end is directed by *Create_Shop_Fixed_Shared_PC*. This means that the transition *Create* will be activated after having received the activating message from *Create_Shop_Fixed_Shared_PC* through PIN1.

PIN2: An input port, one end of which is connected to T1, and the other end is directed by *Building_Distribution_Channel_Inner_PC*, which indicates that T1 will be activated after having received the message of recruiting distributors from *Building_Distribution_Channel_Inner_PC* through PIN2.

PIN3: An input port, one end of which is connected to T3, and the other end is directed by *Product_Purchase_Customizable_Shared_PC*. PIN3 is included in a dashed box, which means that PIN3 is defined only in *Building_Procurement_ Channel_Customizable_Shared_PC* for physical distributors. Having received the message of finishing procurement transactions from *Book_Purchase_Customizable _Shared_PC* through PIN3, T3 will be activated.

POUT1: An output port, one end of which is connected to T2, and the other end is pointing to *Building_Distribution_Channel_Inner_PC*. This means that after having submit the application for distribution eligibility, T2 will send the reviewing message to *Building_Distribution_Channel_Inner_PC* through POUT1.

POUT2: An output port, one end of which is connected to T4, and the other end is pointing to *Online_Sale_Inner_PC* or *Online_and_Offline_Sale_Inner_PC*. This means that after T4 is executed, it will send the communication message to *Online_Sale_Inner_PC* for online distributors or to *Online_and_Offline_Sale_Inner_PC* for physical distritutors through POUT2.

(3) *Online_Sale_Inner_PC*

The sub-process object of online selling ceramic products (denoted *Online_Sale_Inner_PC*) is an *Inner_PC* defined by online distributors. It is used to sell suppliers' ceramic products for online distributors. Its model diagram is shown in Fig. 6.

In Fig. 6, the message places are marked up from P1 to P6. The transition *Create* is used to create the proclet instance, the transition *End* is used to destroy the proclet instance. There are other transitions marked up from T1 to T5. Their details are described as below.

T1: Buyer purchases ceramic products and create orders online.
T2: Buyer pays to online distributors.
T3: Online distributors confirms receivables and notify suppliers for delivery.
T4: Buyer confirms the receipt and finishes the transaction.
T5: Manage sale orders.

PIN1: An input port, one end of which is connected to the transition *Create*, and the other end is directed by *Building_Procurement_Channel_Customizable_Shared_PC*. This means that the transition *Create* will be activated after having received the activating message from it through PIN1.

POUT1: An output port, its one end is connected to T1, and the other end is pointed to *Product_Purchase_Customizable_Shared_PC*. It indicates that after buyer

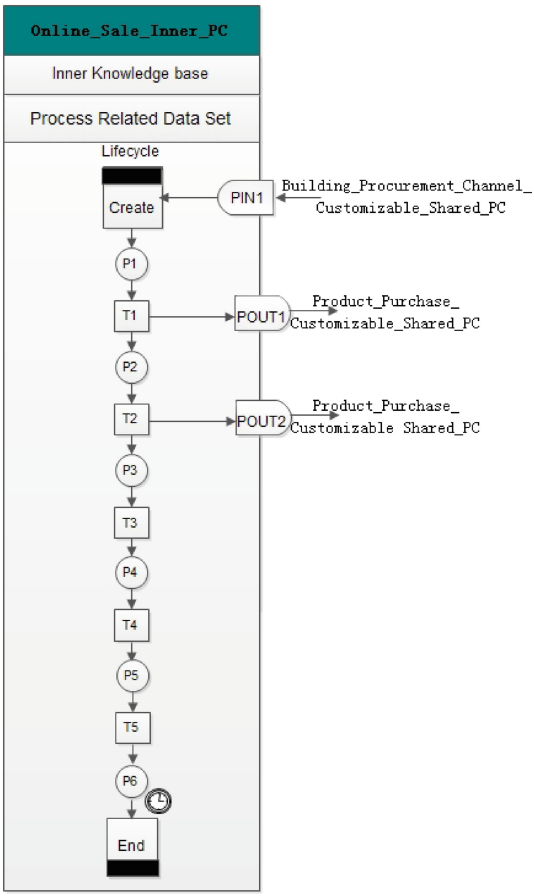

Fig. 6. Model diagram of *Online_Sale_Inner_PC*

created orders, T1 will send the notification message to *Product_Purchase_ Customizable _Shared_PC* through POUT1.

POUT2: An output port, one end of which is connected to T2, another is pointing to *Product_Purchase_Customizable_Shared_PC*. This means that after buyers having pay to the online distributors, T2 will send the notification message to *Product_Purchase_Customizable_Shared_PC* through POUT2.

4.2 Building Interaction Between Proclets

In this section, we model interaction between proclets defined in CDRP. The cloud workflow model built for CDRP is shown in Fig. 7 (to simplify the drawing, the life-cycle of each proclet is omitted, while the input and output ports are depicted).

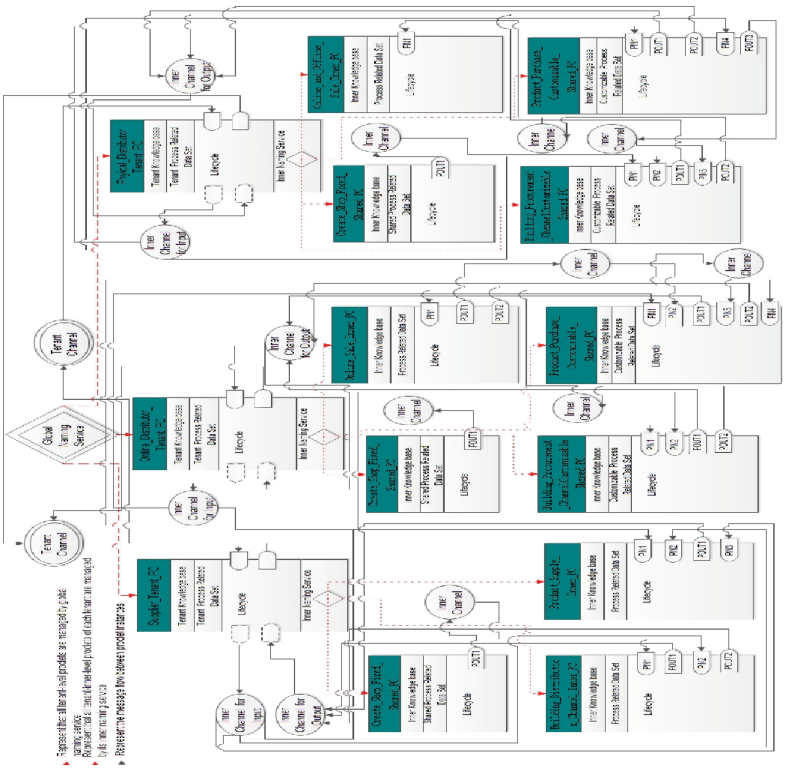

Fig. 7. Model diagram of interaction between proclets

In Fig. 7, the model consists of one global naming service, three tenant-level proclets (respectively denoted as *Suppler_Tenant_PC*, *Online_Distributor_Tenant_PC*, and *Physical_Distributor_Tenant_PC*), some *Inner_PC*s and channels used for interactions between proclets. The working principle of the model involves three aspects: process object reuse, hierarchical management and two-level-channel transmission, the details are given as below.

Process Object Reuse. In CDRP, *Create_Shop_Fixed_Shared_PC* is a *Fixed_Shared_PC*, which can be reused without conducting any modification for three types of tenants. In addtion, *Building_Procurement_Channel_Customizable_Shared_PC* and *Product_Purchase_Customizable_Shared_PC* are *Customizable_Shared_PC*. They can be reused by customizing the transitions and ports of their life-cycle, and configuring the process related data set according to distributors' business requirements.

Hierarchical Management. In Fig. 7, there exists two-level management: tenant-level management and tenant-inner-level management, which can isolate the related private data of tenant business process. In tenant-level management, all tenant-level proclets (i.e. *Suppler_Tenant_PC*, *Online_Distributor_Tenant_PC* and *Phyical Distributor_Tenant_PC*) are monitored and managed by only one global naming service. In tenant-inner-level management, all *Inner_PC*s belonging to a *Tenant_PC* are monitored and

managed by the inner naming service defined in the *Tenant_PC*. For example, in Fig. 7, suppliers' process is composed of three *Inner_PC*s: *Create_Shop_Fixed_Shared_PC*, *Building_Distribution_Channel_Inner_PC* and *Product_Supply_Inner_PC*. They are monitored and managed by the inner naming service defined in *Suppler_Tenant_PC*. This is the same as distributor's process.

Two-level-channel Transmission. In Fig. 7, there exist two types of transmissions: transmission through tenant-inner-level channels and transmission through both tenant-level channels and tenant-inner-level channels. Interaction between tenant-inner-level proclets belonging to the same tenant-level proclet is achieved by the first transmission. For example, in supplier tenant's business process, *Create_Shop_Fixed_Shared_PC* sends an activating message to *Building_Distribution_Channel_Inner_PC* through the tenant-inner-level channels. To achieve the interaction between tenant-inner-level proclets belonging to different tenant-level proclets, the second transmission is adopted. For instance, in Fig. 7, the process of T3 in suppliers' *Building_Distribution_Channel_Inner_PC* sending a message of recruiting distributors to T1 in online distributors' *Building_Procurement_Channel_Customizable_Shared_PC* is shown as follow. Firstly, the message is sent by the POUT1 connected to T3, and transmitted to the transition *Output Processing* of *Suppler Tenant_PC* for security handling (e.g., security coding, data encryption, etc.) through the tenant-inner-level output channel. Then, the message is transmitted to the transition *Input Processing* of *Online_Distributor Distributor Tenant_PC* for security handling (e.g., security authentication, ciphertext decryption, etc.) through the tenant-level channel. Finally, the message is transmitted to PIN2 through the tenant-inner-level input channel, and T1 is fired after receiving the message. Now, one interaction between *Building_Distribution_Channel_Inner_PC* and *Building_Procurement_Channel_Customizable_Shared_PC* is successfully achieved without disclosing their process related private data.

The above proposed case exeplifies that our modeling framework can explicitly model the interactions between multi-tenant business processes while protecting their private data and reusing common process fragments.

5 Conclusion

In this paper, we extend the proclets approach based on hierarchical management ideology and propose a cloud workflow modeling framework using extended proclets. This framework is successfully applied for designing and implementing the CDRP, which can be accessed by http://www.ccmall.cn. Compared with the existing cloud-based workflow modeling approaches, our approach has the following characteristics:

(1) In most existing cloud workflow modeling approaches, a tenant's business process is modeled into a big and complex process model, which has too many tasks and too complex control logic. Therefore, it is difficult to execute and monitor. Meanwhile it may increase the error probability of process executing. In our framework, each tenant's business process is decomposed into several sub-process objects according to its sub-goals or key artifacts, then these sub-processes are abstracted and packaged into tenant-inner-level proclets. During the modeling process, common sub-processes

belonging to multiple tenants are modeled as *Fixed_Shared_PCs* or *Customizable_Shared_PCs*. It can greatly improve the efficiency of business process modeling for each tenant in cloud workflow system.

(2) The existing cloud workflow modeling frameworks do not support the isolation and protection of the private data related to tenant business processes. In our framework, it can isolate and protect the related private data of tenant business processes by hierarchical modeling and management.

(3) The existing cloud workflow modeling approaches do not model interaction between multi-tenant business processes. In this paper, we adopt two-level-channel transmission to explicitly model interactions between multi-tenant business processes.

In summary, the proposed modeling framework using extended proclets is feasible to model interactions between multi-tenant business processes without disclosing their private data. It also provides a model for designing cloud workflow engine.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No. 61170026, 61100017, the National Science and Technology Ministry of China under Grant Nos. 2012BAH25F02, 2013BAF02B01 and the Fundamental Research Funds for the Central Universities of China under Grant No. 2012211020203, 2042014kf0237.

References

1. Chen Kang, S., Wei-min, Z.: Cloud computing: system instances and current research. J. Softw. **20**(5), 1337–1348 (2009)
2. Deng-Guo, F., Min, Z., Yan, Z., Zhen., X.: Study on cloud computing security. J. Softw. **22**(1), 71–83 (2011)
3. Redding, G., Dumas, M., et al.: A flexible object-centric approach for business process modeling. SOCA **4**(3), 191–201 (2010)
4. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
5. van der Aalst, W.M.P., Mans, R.S., Russell, N.C.: Workflow support using proclets: divide, interact, and conquer. In: Proceedings of IEEE Data Engineering Bullet, pp.16–22 (2009)
6. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: a framework for lightweight interacting workflow processes. Int. J. Coop. Inf. Syst. **10**(4), 443–481 (2001)
7. Mietzner, R., Leymann, F.: Generation of BPEL customization processes for SaaS applications from variability descriptors. In: Proceedings of the 2008 International Conferences on Services Computing (SCC 2008), pp. 359–366 (2008)
8. Michiel, K., Chang-ai, S., Marco, S., Paris, A.: VxBPEL: supporting variability for web services in BPEL. Inf. Softw. Technol. **51**, 258–269 (2009)
9. Ralph, M.: Using Variability Descriptors to Describe Customizable SaaS Application Templates, pp. l–27. Institute of Architecture of Application Systems, 18 January 2008
10. Ralph, M., Andreas, M., Frank, L., Klaus, P.: Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In: Proceeding of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS 2009), pp.18–25 (2009)

http://www.springer.com/978-3-319-19508-7

Asia Pacific Business Process Management
Third Asia Pacific Conference, AP-BPM 2015, Busan, South Korea, June 24-26, 2015, Proceedings
Bae, J.; Suriadi, S.; Wen, L. (Eds.)
2015, XII, 199 p. 84 illus., Softcover
ISBN: 978-3-319-19508-7