A method of demand-driven and data-centric Web service configuration for flexible business process implementation

Xu, B., Xu, L. D., Fei, X., Jiang, L., Cai, H., Wang, S.

Original citation & hyperlink:
Xu, B, Xu, LD, Fei, X, Jiang, L, Cai, H & Wang, S 2016, 'A method of demand-driven and data-centric Web service configuration for flexible business process implementation' Enterprise Information Systems, vol (In Press). DOI: 10.1080/17517575.2016.1150522
https://dx.doi.org/10.1080/17517575.2016.1150522

DOI 10.1080/17517575.2016.1150522
ISSN 1751-7575
ESSN 1751-7583

Publisher: Taylor & Francis

This is an Accepted Manuscript of an article published by Taylor & Francis in Enterprise Information Systems on 3 Mar 2016, available online: http://www.tandfonline.com/10.1080/17517575.2016.115052

A Method of Demand-driven and Data-centric Web Service Configuration for Flexible Business Process Implementation

Boyi Xu; Li Da Xu
College of Economics and Management; Shanghai Jiao Tong University; Shanghai; China
byxu@sjtu.edu.cn; LXu@odu.edu
Xiang Fei
Faculty of Engineering and Computing; Coventry University; Coventry; UK
aa5861@coventry.ac.uk
Lihong Jiang; Hongming Cai; Shuai Wang
School of Software; Shanghai Jiao Tong University; Shanghai; China
jianglh@sjtu.edu.cn;hmcai@sjtu.edu.cn
Corresponding author:Lihong Jiang
[Corresponding Author's Email] Jianglh@sjtu.edu.cn

A Method of Demand-driven and Data-centric Web Service Configuration for Flexible Business Process Implementation

Facing the rapidly changing business environments, implementation of flexible business process is crucial, but difficult especially in data intensive application areas. This study aims to provide scalable and easily accessible information resources to leverage business process management. In this paper, with resource-oriented approach, enterprise data resources are represented as data-centric Web services, grouped on-demand of business requirement, and configured dynamically to adapt to changing business processes. Firstly, a configurable architecture CIRPA involving information resource pool is proposed to act as a scalable and dynamic platform to virtualize enterprise information resources as data-centric Web services. By exposing data-centric resources as REST services in larger granularities, tenant-isolated information resources could be accessed in business process execution. Secondly, dynamic information resource pool is designed to fulfil configurable and on-demand data accessing in business process execution. CIRPA also isolates transaction data from business process while supporting diverse business processes composition. Finally, a case study of using our method in logistics application shows that CIRPA provides an enhanced performance both in static service encapsulation and dynamic service execution in cloud computing environment.

Keywords: Enterprise data management, Business process execution, Cloud computing, Data-centric Web service, Information resource pool; RESTful service

1. Introduction

With the rapid development of information technology and the wide use of enterprise information systems, architectures of enterprise information systems have the trend of transforming from traditional on-premise computing platform into cloud computing environments. The cloud computing facilitates enterprise IT infrastructure in a more flexible and adaptive pattern to improve the availability and affordability of on-demand enterprise resources[29]. Especially, as one of the three typical cloud computing models, Software-as-a-service (SaaS) defines business processes as Web services and addresses the inter-organizational business process interoperability through Web services composition. SaaS is expected to be adopted by most small and medium enterprises (SEMs). Gartner [27] predicts that the public cloud services in the mature APAC region will reach no less than $7.4 billion in 2015, and by 2018, the total cloud services spending in Asia Pacific and Japan will hit $11.5 billion. However, coupled with the improvements in the flexibility of demand-driven business processes management in cloud computing, the complexity of infor-

mation resource accessing during business process execution also increases, especially for data-centric application systems. For example, during the project of the logistic cloud computing platform for Fujian province in China, over 80% logistic companies in Fujian are supposed to deploy their business processes and transaction data through Internet by 2018[28]. In addition to the security issues of the transaction data for each company, it is also challenging for the developers to set up on-demand data access services for these different companies to achieve inter-company co-operations in the supply chain through the platform.

In this paper, we take the business process of handling customer orders in a manufacture industry as an example to explain the challenge we faced in the data-centric Web application development on a cloud computing platform. In manufacturing industries, customer orders have some common activities, such as receiving customer orders, checking product inventory and arranging product manufacture. So these common activities could be carried out by Web services with similar functions. However, for different companies, business processes and transaction data may be slightly different. This leads to the requirement of dynamic configurations and customization of data centric services developments[30].

In summary, in traditional data-centric systems such as ERP systems, data resources, which are stored in relational databases or in formatted documents, are not only used for storing and displaying, but also used for data analysis and data mining for business purposes. In cloud environment, it is more challenging to provide a scalable and flexible transaction data display or data analysis. This is because how to construct data-centric services for related information resources across different tenants so as to maximize resource utilization and minimize development cost, while keeping multi-tenant resources isolation for the purpose data privacy is not an easy task [1,2].

Therefore, we propose a Configurable Information Resource Pool Architecture (CIRPA) so as to realize an effective business process execution for multi-tenants in cloud environment. CIRPA focuses on the representation and management of data resources in a configurable and scalable way of data-centric Web service development.

The remainder of this paper is organized as follows. Related works is overviewed in section2. In section 3, the overall framework is introduced. In section 4, detailed design and implementations of CIRPA are shown. Then we present a case study in section 5. Section 6 concludes this paper.

2. Related works

In order to implement on-demand data services through cloud computing, more researches on SOA-based BPM systems have shifted their focuses from seeking the accuracy and QoS criteria of service composition to fulfilling the flexibility and configurability of the business process execution, especially in data-centric environment [3] [4].

In cloud-based applications, transaction data isolation among different unites and organizations has been recognized as the first of all problems that need to be solved by developers. Researchers in [5] review the challenges in collaborative business process management to support information flowing among organizations seamlessly, which include business process modeling, privacy and confidentiality, and process evolvement management. Service-oriented architecture and pervasive computing are summarized as two most typical technologies to enhance the ability of IT systems' collaborative business process management where trust and security issues are critical. The paper [6] proposes a system

architecture for telecommunication corporations to integrate heterogeneous platforms and provide diverse available e-services based on SOA to customers. Several layers are divided as the presentation layer, the interaction layer, the exchange layer, the processing layer and the data layer. Multi-tier models could facilitate the transmission of customer information among different business processes. However the authors think that the agility and integration in operation system level still need to be improved to support the e-services integration in communication industry.

In data-centric applications, the problem of flexible data integration become more challenging with the wide use of cloud computing. REST(Representation State Transfer) architectural style proposed by Roy Fielding [25] is widely used to construct data centric Web services, focusing on the standard data operations like *GET*, *PUT*, *POST* and *DELET* through the Internet. RESTful-based approaches blossom in cloud information analysis especially for IOT (Internet of Things) applications[26]. Because data-centric Web services usually have similar operations on data or information resources, automatic generation of web services is considered to be more efficient than writing code manually. Based on this idea, an open resource-based integrated platform system [7] is proposed based on ROA (Resource-Oriented Architecture). It uses ontology to organize and manage enterprise information resources. Furthermore, on the basis of the Resource Linking Language [8], a meta-model is provided for descriptions of RESTful services so as to carry out service documentation and composition. Accountable State Transfer (AST) architecture [9] is proposed to bridge the accountability gap in REST, which provides accountability capabilities for the service execution.

In addition to the methods of automatic generation of RESTful services, automatic service composition methods are also considered to be effective to improve the ability of data manipulation in cloud computing [10][11]. Therefore, service agents are explored and used in [11] to improve the scalability of service composition in a distributed approach. In [10], an architecture combining static service composition and dynamic service composition which includes stages of workflow planning, services discovering, services selecting, and services executing, is proposed to realize seamless workflow orchestration and scalable functional integration. Although automatic service composition is too complex to implement fully, semantic technology, such as ontology which is illustrated in this research, shows promising view of developing more flexible architecture for dynamic business process management, given that the Web services could be encapsulated and notated in proper ways which could be recognized and allocated on premise during service composition.

Together with the research of service composition methods in cloud computing, amounts of cloud-computing-based BPM systems are put forward to enhance the scalability and configurability of business process management with cloud computing [12] [13] [14]. Research in [1] introduces the developments and challenges in workflow management in cloud computing infrastructures, and put forth a solution based on the BeesyCluster middleware for distributed management of services with static and dynamic rescheduling within a market of services. Paper [15] presents a multi-tenant workflow engine that enables different tenants to run their workflows securely within the same workflow engine instance. Moreover, in the paper [16], a cloud-based model driven development and execution environment provides a shared business process modeling workspace and a business process execution environment. However, in research [16], resources allocation method is not involved in both the period of business process modeling and the service execution.

Although one of the significant ideas in cloud computing is that functions of the information systems could be simply plugged and then played bringing high scalability and

flexibility in business process interoperations, performances need to be traded off against challenges in data security and interoperability when enterprises decide to adopt cloud computing platform to organize their business processes [17]. Thus, on the one hand, agent-based method is explored to facilitate service discovering and service negotiation in cloud commerce searching engine for Web service composition [20]. On the other hand, platforms integrating applications deployed in public clouds with intra information systems to improve business process performances are also researched to protect enterprise data security under cloud computing environment [19].

For cloud service composition, evolutionary algorithms, fuzzy logic and ontology-based approach are used to analyze the compatibility of services provided in clouds, and to optimize the services composition. Scalability, fault tolerance, heterogeneity and large scale data management are considered as challenging problems in data-intensive cloud computing applications [21]. Actually, even in process-driven and service-oriented architectures, with the increasing of processes accumulation in cloud platforms, data accessing becomes more complicated than the traditional enterprise information systems. In order to effectively support data accessing in business processes, [22] proposes a View-based Data Modeling Framework to trace data accessing activities in business processes by data access activities specification, integration, and extraction. Paper [23] discusses the issue of service specification using models which capture the scope, capability and state of a service. The models make integration of services more certain and allow previously incompatible services to be combined easily. The authors prototyped Web2Exchange, a platform for modeling, transforming and integrating services. Hibernate and EJB plus Spring are popular approaches used in practice for representing and accessing persistence objects such as entity bean [24], but are relatively weak in performance of information resource scalability and lifecycle control. We will compare Web2Exchange and the method of Spring plus Hibernate with our CIRSP in section 5.2.

All of the above-mentioned works demonstrate that enterprise system's migration towards the cloud is attracting both researchers and practitioners. The characteristics offered by the state-of-the-art cloud computing technology, including flexibility, on-demand, accessibility, are studied and utilized intensively. On the other hand, the mentioned works also show that enterprise system's migration towards the cloud is still in its infancy, especially from the viewpoint of business demand instead of technology push. In addition, the multi-tenant service isolation lacks business consideration. Thus these methods will be low efficient to fulfill different types of applications especially data-centric systems. It is a promising way to execute enterprise applications by means of allocating or encapsulating data resources according to business features so as to build "inherent tenant-isolated" services in cloud environment where service encapsulations need to be tackled.

Next section presents the design of CIRPA framework, from which we tried to demonstrate how to realize scalability and flexibility for data-centric services in business level. We hope to shed new light on the design of basic building blocks for cloud-based business processes execution

3. Architecture of configurable information resource pool

The main goal of the design of the Configurable Information Resource Pool Architecture (CIRPA) is to realize data encapsulation and management in business process level for flexible business processes execution. Fig.1 shows the framework of CIRPA considering scalable and dynamic information resources aggregation in business process level. In CIRPA,

Information Resource Pool is a container of data-centric services. These data-centric services feature bigger granularity and are automatically generated to manipulate information resources while a business process is executed. When business process instances are executed, information resource pools are constructed dynamically; and the information resources are added into the pools by data-centric SOA services according to the definitions in *Reference Resource* models. A *Reference Resource* model is a conceptual level description of data stored in either relational databases or unstructured files in cloud environment. Each *Reference Resource* model functions not only as a bridge between physical data entities in data sources and virtual information resources, but also as the start points to generate RESTful services which manipulate the information resources during business processes execution.

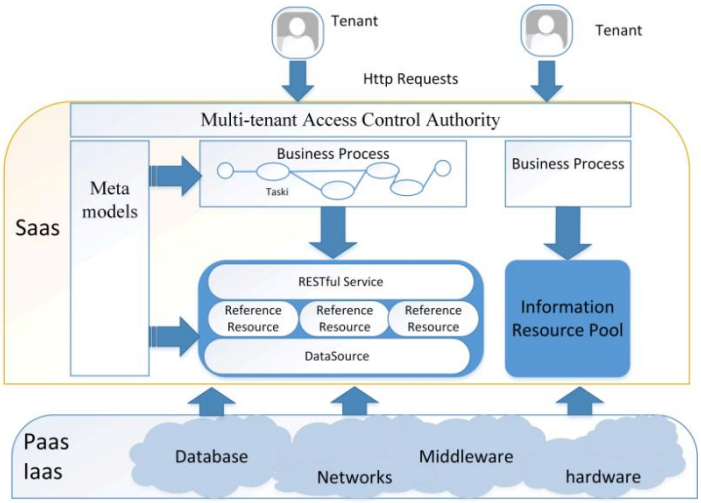

Fig. 1. CIRPA architecture for dynamic service generation and execution in cloud environment

CIRPA essentially consists of two parts, static module and dynamic module. Static modules mainly support model mapping in design stage, while dynamic modules achieve on demand bindings in runtime stage.

Static modules concentrate on the static aspects, for instance, description and representation of resources and relations between them, and expose resources as RESTful services. With static modules, CIRPA provides configurability and integrality in service design stages.

The dynamic modules are in charge of the lifecycle of resources, offering safety control and performance optimization in service execution stages. Dynamic management of information resource pool involves the state-space construction based on the resource list of a certain business process (for details on the state-space construction, please refer to Section 4.4). The lifecycle of information resource pool is synchronous to the business process execution so as to realize scalable and dynamic data configuration in cloud environment.

CIRPA static modules are defined by meta-models. Meta-models determine both the definition of business processes and the extraction of reference resources so as to enable the configuration of distributed data sources.

4. Detailed design of configurable information resource pool

In this section, detailed design of CIRPA is presented that fulfils configurable and on-demand data accessing for dynamically composed business process execution in cloud computing. Further, the main steps that facilitate the application of the information resource pool are also provided.

4.1 Information metemodel definition

Prior to service construction, it is essential to define a unified architecture for information modeling. Based on multi-view business modeling, a meta-model for business process execution is shown in Fig.2. Three types of models are defined, including process model, resource model and organization model.

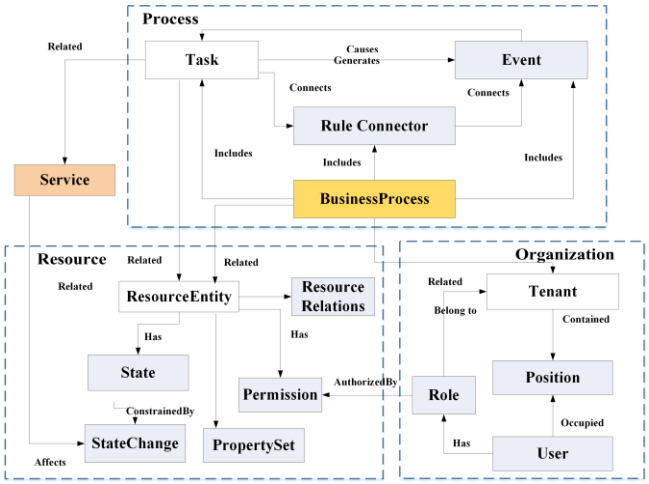

Fig. 2. Metamodel for business process execution

Definition 1: *Metamodel of Business Process (MBP)*

<MBP > = {*Tenant, Tasks, Events, Connectors, Arc* } (1)

In (1), *Tenant* represents related organization unit;

Tasks are the descriptions of services and data entities;

Connectors={AND,XOR}+;

Arc ={Connector→(Event,Task),(Event,Task)→Connector};

*MBP*s are not only the descriptions of the control-flows of business processes, but also the descriptions of the relationships between resource entities and services for further disposing.

Definition 2: *Tenant Model*

The *Tenant Model* is an organizational unit, which is defined as a set of *Positions*, *Roles*, and *Relationships* between positions and roles.

Tenant = <Positions, Roles, Relationship> (2)

Definition3: *ResourceEntity Model*

ResourceEntity Model is the abstraction of business data. To enable configurable and on-demand data accessing, tables in databases can be divided into smaller resources, and further, various resources can be aggregated to bigger resources to achieve dynamic business process execution.

Resource = (URI, RPS, ROS, RRelation, RState) (3)

Each *Resource* has a unique URI to identify itself.
RPS is the set of resource properties.
ROS defines *RESTful Get/Put/Post/Delete* operations invoked via *http* requests.
RRelation represents resources relations between two resource entities.
RStates describes the states of a resource during execution.

4.2 Mapping mechanism from process model to resource model

In order to provide persistent and consistent data accessing during a business process execution, for a given business process, resources extraction is carried out to gain a list of related information resources, denoted as *ResourceList*. The algorithm for finding the list of related information resources is described in a pseudo code segment as shown in fig. 3.

```
Get mMBP {ProcessID};
If (∀S∈{ Tasks of ProcessID },noEmpty(S)==true) then do
   Create RelatedResourceList {ResourceList};
   For (s:S) do begin
      A:=s.getTasksList();
      For (a:A) do begin
          if (a.value==null)
          else if (isReferenced(a)==true) then
               Search (ReferenceResource(a));
               AddResourceID ();
          end else if;
      end For;
  end For;
  MergeSameResource {ResourceList }
  PermissionCheck {ResourceList }
Else print" {S|nonEmpty(S)=false, S∈{ mMBP }} no resources atained";
End if;
```

Fig. 3. Algorithm for finding related information resources

In Fig.3, variables of each task in the business process are searched in the resources references. If a resource and a variable share the same/similar attributes or the same states, the resource will be added in the resource list. After all mapped resources are added in the resource list, *MergeSameRsource{RosourceList}* is then executed to find and merge synonym resources according to the semantic notation of resources. *PermissionCheck{ResourceList}* checks the permission for each resource so that only permitted resources could be included in the resource list for further accessing by the business processes.

Table 1 illustrates the mapping from business processes to information resources.

Table 1. Mapping from business processes to information resources

Source Model	Target Model	Reference Argument	Relationship Type
Process Model	Function Model	$Act \subseteq FM^{+}$	One-to-One
Process Model	Tenant Model	Tenant Model $\subseteq$ RoleSet	One-to-One
Function Model	Information Resource	$ParaType \subseteq IR^{*}$	One-to-One
Tenant Model	Information Resource	1.{Tenant1.getRoleList()} $\subseteq$ Role 2{$role_1$.getPermissionList()} $\subseteq$ Permission	One-to-Many One-to-Many

		3.{permission$_1$. getResourceList()} $\subseteq$ Resource	One-to-Many
Information Resource	ResourceList	1.$\forall$attribute$_1$, attribute$_2 \in$ ResAttr Res(attribute$_1$)=Res(attribute$_2$) $\in$ ResID	One-to-Many
		2.$\forall$state $_1$, state$_2 \in$ ResState Res(state $_1$)=Res(state $_2$) $\in$ ResID	One-to-Many
		3.$\forall$attribute $\in$ ResAttr ResAttrType(attribute) $\in$ {SimpleType} $\cup$ {ResID}	One-to-one

The detailed mapping mechanisms from business processes to information resources are described as follows:

- FM^+ represents the transitive closure of the tasks in a function model. *Act* is a subset of FM^+ and denotes the activities derived and directly mapped from a business process.
- RoleSet represents the whole set of roles in an organization model. TenantModel is a subset of RoleSet and denotes roles involved in a business process. During the period of business process design, roles will be mapped into a tenant model. In a tenant model, one tenant could be assigned with multiple roles by {*Tenant1.getRoleList()*}, and likewise one role could be assigned with multiple manipulation permissions for multiple resources by {role1.getPermissionList()} and {permission1. getResourceList()} respectively.
- IR^* represents the group of units of information resources in information resource models. The information resource model is created by corresponding process model when a business process is designed, accessed by the business process in runtime, and deleted after the business process has been executed.
- ParaType is a subset of IR^* and denotes the definition of a set of resources for a function model. During the mapping process from Information Resource to resource list, a resource will be added in the resource list if either of the following conditions is satisfied.
  1. Res(attribute$_1$)=Res(attribute$_2$)$\in$ResID
  2. Res(state$_1$)=Res(state$_2$)$\in$ResID
  3. ResAttrType(attribute) $\in${SimpleType}$\cup${ResID}

4.3 Mapping mechanism from resource lists to RESTful services

After related *ResourceList* of a certain process is constructed, we configure a composed resource to bind to the data sources and also we generate *RESTful* services for applications.

In order to facilitate the configuration of the information resources, we use an extendable, unified and easily understandable way to represent the resources and their relations via XML format. Fig.4 gives an XML file example of a composed resource.

```
<resource name="schoolWithRelations" table="school" isCompound="true">
        <id name="id" type="java.lang.Integer">
                <column name="id" not-null="true"/>
                <generator class="identity"/>
        </id>
        <property name="school name" type="java.lang.String">
                <column name="name" not-null="false"/>
        </property>
        <property name="detail address" type="java.lang.String">
                <column name="address" not-null="true"/>
        </property>
        <relationproperty name="students" >
                <relation refid="has_student" />
        </relationproperty>

        <relations>
                <relation type="oripsResourcePool.relations.has" id="has_student" name="relation between school and students">
                        <resource refname="studentWithRelations">
                                <refid name="school_id"/>
                        </resource>
                </relation>
        </relations>
</resource>
```

Fig. 4. An XML example of composed resource

In order to precisely abstract and represent the relations among the resources in the real world, we define the types of resource relations following the ontology model. There are four basic types of relations in ontology: "*part of*" represents relation between the part and the whole, "*kind of*" is an inheritance relation, "*instance of*" represents the relation between instances and classes, "*attribute of*" is an aggregation relation. We implement these four relation types in CIRPA. Thus, each resource refers to either one table or recursively other resources. For instance, the resource "*schoolWithRelations*" refers to the table "*school*". If the property "*isCompound*" of the resource "*schoolWithRelations*" is true, it means that the resource "*schoolWithRelations*" is a composed resource and has relations with other resources. The properties are the core part of resource, each of which represents either a table column or a resource property. The property "*relations*" of "*schoolWithRelations*" is none-empty if it has a relation with other resources such as "*studentWithRelations*". The relation type "*attribute of*" means that "*schoolWithRelations*" aggregates one or many "*studentWithRelation*" instances.

After composed resources are constructed, RESTful service could be generated to encapsulate and expose the related information resources to Web applications

Ref [7] gives a detailed description of how RESTful services are generated. Given the RESTful services, CIRPA creates different WADL files to provide accessing and operating interfaces of various granularities.

4.4 Lifecycle management of information resource pool

Lifecycle management of information resource pool is constructed to fulfill on-demand service execution. Based on the states of the resources in the *ResourceList* and the state transition diagram, *CIRPA* dynamically manage resources according to the resource states. We define six states in *CIRPA* which are *created, runnable, waited, deleted, aggregated,* and *separated*. The *created*, *runnable*, *waited* and *deleted* states are basic states which represent the lifecycle of resources. *Aggregated* state and *separated* state are two special states which manifest relations between multiple resources.

***Created* state:** When a resource is initialized and defined, it will be in the *created* state, and *CIRPA* will record the created time in the log module. A *created* resource still cannot be accessed via http request. It needs to wait for the notification for further processing.

***Runnable* state:** A *runnable* resource means one or many requests are accessing this resource. *Runnable* resource collaborates closely with the *Access Queue* and *Database Security Control Module*. If a resource is accessed frequently, *CIRPA* will put it into cache for better performance such as lower I/O operation time.

***Waited* state:** A *created* resource or a *runnable* resource can be transited into a *waited* resource which cannot be requested for further operations. For example, if the service granularity for a *runnable* resource is too large to satisfy the functional requirements, the resource state will be set to *waited*. After additional resource relations are added, it will be reset to *runnable*.

***Deleted* state:** If a resource is not needed any more or if it has technical failure, it will be set to *deleted* state. For instance, *CIRPA* checks the last operation time of resources at pre-set interval. If the time between the last operation time and current time exceeds the pre-set value, this resource will be deleted and its state will be set to *deleted*.

***Aggregated* state and *separated* state:** If a resource is in *aggregated* state or *separated* state, its corresponding relations with other resources could be built by *CIRPA*. *Aggregated* resource means that this resource is a property of another resource. *Separated* resource means that this resource can be divided into multiple resources of smaller granularity. The algorithms of resource aggregation and separation are shown in Fig.5. With functions of resource aggregation and resource separation, *CIRPA* can provide scalable resources configuration to meet the demands on data resources of various granularities.

```
Program AggregateResource ()
  //Get Aggregated Resource List
  ResourceList=GetAggregatedResourceList();
  For all Resource in ResourceList
    //Get the Relation Property of this Resource
    RelationProp = GetRelationProperties(Resource);
    For all Res in ResourcePool
      //Find a resource that can be aggregated
        If (hasRelation(Res, RelationProp)) //Build Relation
        addRelation(Res,Resource,RelationProp.TYPE);

Program SeparateResource ()
  //Get Separated Resource List
  ResourceList=GetSeparatedResourceList();
  For all Resource in ResourceList
    //Get the Separated Property of this Resource
    SeparatedProps = GetSeparatedProperties(Resource);
    For all prop in SeparatedProps
      //Get the Sub Properties
      SubProps = GetSubProperties(Resource, prop);
      //Generate Sub Resource and WADL File
      SubResource = GenerateResource(SubProps);
      GenerateWADL(SubResource);
      Delete (Resource); //Delete the Original Resource
```

Fig. 5. Algorithm of aggregating and separating resource

In algorithm *AggregateResouce*, if a resource *Res* is determined to have relation with another resource in the *ResourceList* by function of *hasRelation()*, these two resources are then aggregated by adding "*attribute of*" relation between them, to form a new larger resource.

On the other hand, in algorithm *SeparateResource*, a resource could be divided according to its properties. Each property could then generate a new smaller resource.

Using algorithms of *AggregateResouce* and *SeparateResource*, CIRPA changes the information resource to either larger or smaller size to support the business process composition or decomposition.

Together with resource operating algorithms and different resource states mentioned above, CIRPA implements dynamic resource lifecycle management to provide configurable data accessing in business process execution.

5. Case study and discussion

A case study of logistics process is provided for verification, and related discussion is also given.

5.1 Case study

In this section, we take a production logistics process as an example and use *CIRPA* platform to provide flexible information resource accessing for dynamic business processes composition.

The basic requirements of the firm in this case study are as follows:

- The firm intends to build a data centric backend system for lower layer data management and upper layer data applications.
- The firm has heterogeneous data resources: some from ORACLE databases, and others from MYSQL databases.
- Due to the access control, same data resources may be displayed in different views to people of various authorities.
- The firm needs to configure resources dynamically through dividing or aggregating resources based on certain business requirements.
- Production logistics services need to be provided with various granularities in high concurrency environment.

The whole process of using *CIRPA* to dynamically configure information resources according to the business processes is introduced by steps as follows.

Step1： Process Modeling

The process begins when an order is received, and then the sales departments check the product inventory. If the product inventory is not enough for the order, a production plan is created and sent to a production department. The production department checks the material inventory. If the material inventory is not enough for the plan, a purchase order is created and sent to a purchasing department. The purchasing department will search supplier for materials. The process waits for the purchase to be finished. After obtaining the materials, the production departments manufacture the products, package them and send them to product inventory. Then the sales departments will send the order to a dispatching department, which will deliver the cargo to the customer. After the customer confirms receipt, the whole process is finished. The BPMN (Business Process Modeling Notation) diagram is given, as Fig.6 shows.

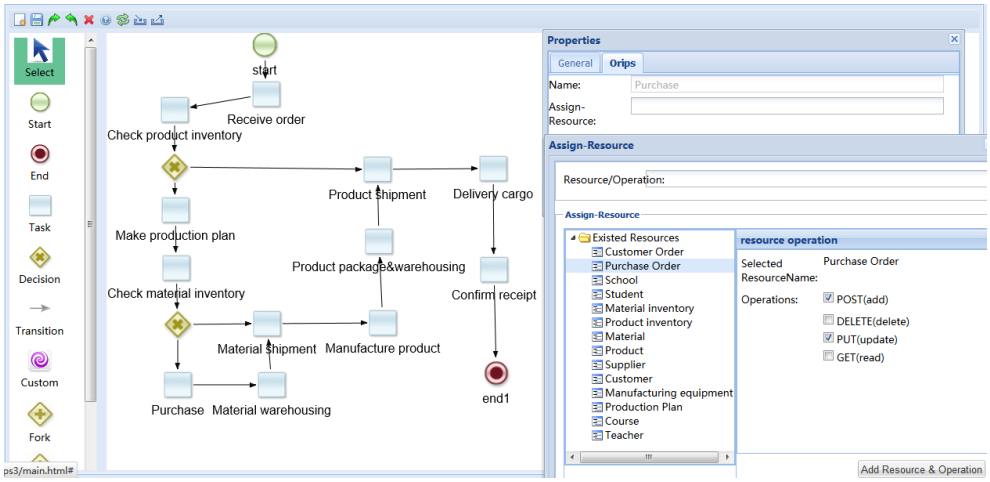

Fig. 6. A business process of logistics application

Fig.6 consists of two parts. The left half illustrates the business process design. Design engineers can make use of the tools in the tool bar to construct personalized processes for specific tenants. The right part lists the corresponding resources and their possible operations for each activity in the process. The novelty of this paper is that CIRPA enables flexible business process implementation by providing a platform so that designers can choose the resources and their operations conveniently; and further, RESTful operations can be generated on-demand based on pre-built templates.

Step 2: Resources Mapping

After process modeling, we analyze the resources involved in the process and map them to activities. Resource and activity relations are given in Table 2.

Table 2. Activity/Resource Mapping

Activity	**{Related Resource/Operations}**
Receive order	{Customer order/POST}
Check product inventory	{Customer order/GET,PUT},{Product inventory/GET}
Make production plan	{Production plan/POST}
Check material inventory	{Production plan/GET,PUT},{Material inventory/GET,PUT}
purchase	{Purchase order/POST,PUT},{Supplier/GET}
Material warehouse	{Purchase order/DELETE}{Material/POST}, {Material inventory/PUT}
Material shipment	{Material/PUT},{Material inventory/PUT},{Production plan/PUT }
Manufacture product	{Material/DELETE},{Product/POST},{Manufacturing equipment/GET,PUT},{Production plan/PUT,DELETE}
Product package & warehouse	{Product/PUT},{Product inventory/PUT},{Customer order/PUT}
Product shipment	{Product/PUT},{Product inventory/PUT}
Delivery cargo	{Product inventory/PUT},{Carrier vehicle/GET,PUT}
Confirm receipt	{Customer order/GET,PUT},{Product/PUT}

Step3： Resource Configuration

Resource relations can be added through a configurable way to provide compound resources of different granularities. For example, "*Purchase order"* has aggregation relations with "*Supplier"* and "*Material"*.

When assigning relations to resources, CIRPA will change the database table structures dynamically such as adding foreign keys to columns and creating intermediate tables for many-to-many relations.

If special relations such as "*association*" are created, CIRPA will add certain relation description information to the *Resource Relations Library*. For example, *Carrier vehicle* has association relations with *Product*.

All of the resources are contained in the resource pool. Fig.7 shows the details of relations for resource configuration.

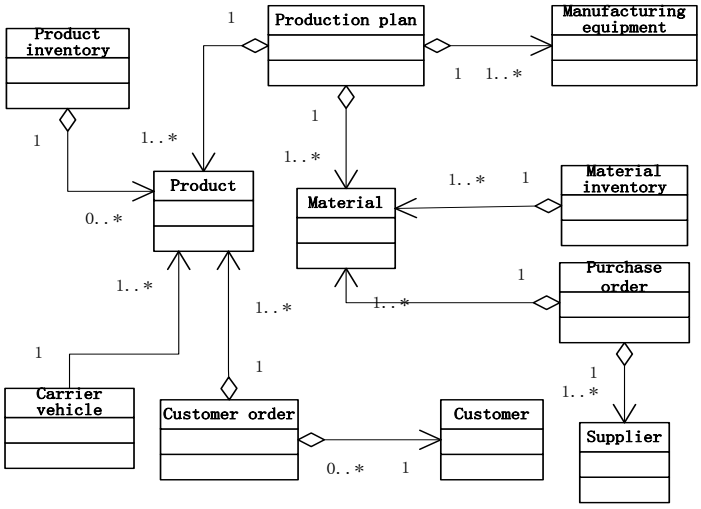

Fig. 7. Relation diagram for resource configuration

Step4: Service Encapsulation

In order to create and model resources, users should bind data to resources via a web user interface provided by CIRPA. The configuration of resource-data mapping includes connecting to database and mapping from tables or their attributes to resources.

After accomplishing the above steps, CIRPA turns into run-time mode. It offers RESTful interfaces to upper layer application systems. For instance, users can send *HTTP* requests to find resources. Table 3 shows the request format and response content.

Table 3. *HTTP* Request and Response

Request: *Get: http://localhost/ResourcePool/Product/price/low=500&high=1000* *Response:* *[{"id":"130",”price”:600,”name”:”computer”,”weight”:”5”},{"id":"146",”price”:800,”name”:”mp3”,”weight”:”1”},{"id":"189",”price”:950,”name”:”phone”,”weight”:”1”}]*

This response returns the products whose prices are between 500 and 1000. We use JSON (JavaScript Object Notation) format to carry out data transmission between web frontends. The CIRPA also supports XML format.

CIRPA adds conditions to RESTful operations, such as “*low=500&high=1000*” in “*Get: http://localhost/ResourcePool/Product/price/low=500&high=1000”* to enhance the capability of URL (Uniform Resource Locator) querying.

If users need a compound resource, they can send a request as shown in Table 4.

Table 4. Compound Resource Request through RESTful Operation

Post: *http://localhost/ResourcePool/Customer order* *Post Content:* *{“id”:”122”,”totalprice”:”1700”,”date”:”20131208”,”customer”:{“id”:”11”,”name”:”andy”,”address”:”SJTU”,”phone”:”16818181688”},”product”:[{"id":"130",”price”:”600”,”name”:”computer”,”weight”:”5”},{"id":"146",”price”:”800”,”name”:”mp3”,”weight”:”1”},{"id":"112",”price”:”300”,”name”:”nokia phone”,”weight”:”2”}]}*

This request generates a resource in bigger granularity named “*Customer order*” which consists of two sub resources named “*customer*” and “*product*” respectively.

In CIRPA, the generated RESTful services are represented in WADL format, as shown in Fig.8.

```
<xs:complexType name="Customer">
  <xs:all>
    <xs:element name="name" minOccurs="1" type="xs:string"/>
    <xs:element name="phone" minOccurs="1" type="xs:string"/>
    <xs:element name="id" type="xs:int" minOccurs="0"/>
    <xs:element name="address" minOccurs="1" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:complexType name="CustomerCollection">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Customer_Instance" nillable="true" type="tns:Customer"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="Customer orderWithRelations">
  <xs:all>
    <xs:element name="totalPrice" type="xs:int" minOccurs="0"/>
    <xs:element name="date" type="xs:int" minOccurs="0"/>
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <xs:element name="id" type="xs:int" minOccurs="0"/>
    <xs:element name="Product" type="tns:ProductCollection" maxOccurs="1"/>
    <xs:element name="Customer" type="tns:CustomerCollection" maxOccurs="1"/>
  </xs:all>
</xs:complexType>
</xs:schema>
</grammars>
<resources base="http://localhost/ResourcePool">
<resource path="/Customer orderWithRelations">
  <method name="GET">
    <request/>
    <response>
      <representation mediaType="application/json" element="prefix1:Customer orderWithRelationsCollection"/>
    </response>
  </method>
  <method name="POST">
    <request>
      <representation mediaType="application/json" element="prefix1:Customer orderWithRelations"/>
    </request>
```

Fig. 8. A generated WADL file of RESTful service

Step5: Resource Pool Management

When business processes are executed, users can monitor the resources states in real time through CIRPA. They can also manage resources without suspending the services. Users can change the granularities of the resources, add resource relations and change resource states. For instance, if a resource is on heavy demand, users can set the state of this resource as *waited* to release the load of servers as shown in Fig.9.

In this case study, when the logistics process starts, the resources are configured based on the business process model. First, activity of "*Receiving order"* creates a "*Customer order"* resource. Then inventory of corresponding product is checked. If the inventory is enough to fulfil the order, "*Customer order"* turns into *runnable* state. The selected products will delete aggregation relations with "*Product inventory"* and setup aggregation relations with the "*Customer order"* being executed. If the inventory of the product is not sufficient to fulfil the customer order, "*Customer order"* and "*Product inventory"* will turn into *waited* state, and wait until enough products are produced by the production departments. The *waited* state of the *"Customer order"* is one of the preconditions of the activity of "*Make production plan"*.

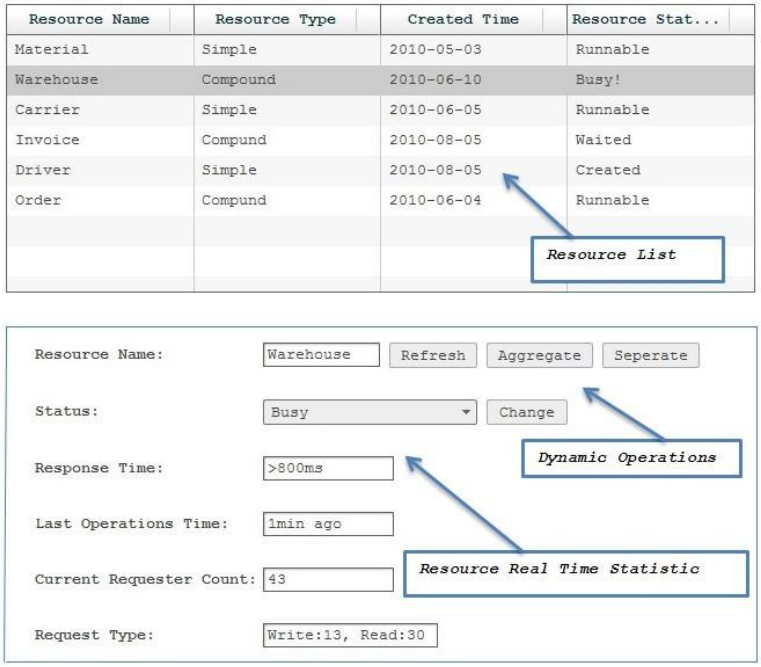

Fig. 9. Resource Pool Management

Table 5 shows a fragment of the resource state transition matrix. In Table 5, *c*, *r*, *w*, *a* , *s* and *d* represent the resource states of *created*, *runnable*, *waited*, *aggregated*, *separated* and *deleted* respectively. *A4*, *A5*, *A6*, *A7* and *A8* represent the business activities of *Check material inventory*, *Purchase*, *Material warehousing*, *Material shipment* and *Manufacture product* respectively.

Table 5. Part of state transition matrix

	A4	**A5**	**A6**	**A7**	**A8**
Material inventory	(r,w)		(w,r)		
Material			(c,r,a,s)	(a,s,a)	(a,d)
Manufacturing equipment					(r,a,r)
product					(c)
Production plan	(r,w)			(w,r)	(r,d)
Purchase order		(c,r,w)	(w,r,d)		

5.2 Discussion

There are other approaches to implement data-centric systems. Spring plus Hibernate is a popular solution among them. Based on Hibernate ORM (Hibernate Object-Relation Mapping), encapsulation of JDBC and the Inversion of Control (IOC) container of Spring, enterprise can easily build up data backend system without many difficulties. Recently, new version of Spring also supports RESTful service as some other approaches do. However, the solution has some disadvantages. Firstly, it doesn't fully support multiple databases connections so that complex data configuration is unavoidable. Secondly, this solution only focuses on the representation of resources; it doesn't provide the monitoring and management of resources which are essential in resource-oriented architectures. Thirdly, in response to the scalability of applications, hand-coding or manual operations are needed which increase the cost of the development.

In general, Spring plus Hibernate is a good solution for common data-centric backend system, but is not a good choice in developing data-centric application in the Cloud environment. Web2Exchange is mainly a model-based service transformation and integration environment [23]. Table 6 presents a comparison of CIRPA with these two approaches.

Compared with other approaches, due to the introduction of meta-model, CIRPA supports larger granularity RESTful service both in static information encapsulation and dynamic Web service management in the cloud environment.

CIRPA provides a scalable and dynamic architecture based on service and data aggregation. By offering RESTful services generated from business processes, information resources will be rapidly integrated, manipulated and exposed for data-centric applications. Information resources are assigned with well-structured URIs so that they can be better managed, with full support of advanced data access control, resource management and resource life cycle control.

CIRPA not only supports a modeling method for data configuration or aggregation in design period, but also provides a control mechanism in service execution period. Based on the state transition matrix defined with the ResourceList extracted from a certain business process, the business goal and constraint could be used to control the whole execution no matter whether the granularity of services is large or small.

One problem of the platform is that CIRPA may produce large number of services when data is aggregated from different model views. Thus ontology is needed to build semantic relationships between resources so as to organize and manage these resources in semantic level.

Table 6. Comparison of CIRPA with other approaches

Features	**Solutions**		
	CIRPA	***Spring + Hibernate***	***Web2Exchange***
Model Element	Data-centric Resources	Persisting Objects	Model-based services
Multiple Database Connections	Fully Support	Manual	Support
Relational Data Mapping	Fully Support	Fully Support	Manual
Data Aggregation and Separation	Fully Support	Manual	Not Support
Data Monitoring	Fully Support	Not Support	Support
Data Lock and Data Cache	Fully Support	Manual	Not Support
Data Transaction	Manual	Manual	Manual
Service Configuration	Automatically	Not Support	Manual Mashup
RESTful Data Accessing	Fully Support	Support	Fully Support
Multi-tenants isolation	Support	Manual	Not Support
Dynamic Runtime Management	Fully Support	Not Support	Manual
Semantics disposing	Support	Not Support	Not Support

6. Conclusion

The paper investigated the current developments in business process management, focusing on challenges that appear especially in data-centric information systems when business processes are composited and executed dynamically across business partners.

The following challenges in the construction of cloud based BPM system were explored, including (1) how to design a scalable information system architecture to leverage

flexible business process management, (2) how to deal with the complexity of cloud-enabled on-demand transaction data accessing for dynamic business process execution, and (3) how to isolate and effectively manage transaction data from business process to support diverse business process composition.

An architecture for data-centric service generation and execution is put forward to support business process interoperation to enhance the flexibility of business processes management in cloud computing. Based on the architecture, a platform is developed to facilitate the process design, resource configuration and Web services generation. The contribution of the paper is that we propose a method and also implement a platform that enables to encapsulate and expose information resources as web services automatically in business level. A case study in manufacturing industry is also provided to demonstrate and verify the effectiveness and usefulness of CIRPA.

Further research work will be carried out based on ontology construction and evolution to adapt to large-scale data management and intelligent application implementation.

References

1. Pawel Czarnul, Integration of cloud-based services into distributed workflow systems: Challenges and solutions, Scalable Computing: Practice and Experience,Volume 13, Number 4, (2012): 325–338.
2. Jie Zhu, Bo Gao, Zhihu Wang, Berthold Reinwald, Changjie Guo, Xiaoping Li, Wei Sun, A Dynamic Resource Allocation Algorithm for Database-as-a-Service, 2011 IEEE International Conference on Web Services(ICWS2011), IEEE Press, 2011, pp564-571, DOI 10.1109/ICWS.2011.64.
3. Giorgio Bruno, Frank Dengler, Ben Jennings, et.al., Key challenges for enabling agile BPM with social software, Journal of Software Maintenance and Evolution,2011; 23:297–326 DOI: 10.1002/smr.523.
4. Saoussen Cheikhrouhou, Slim Kallel, Nawal Guermouche, Mohamed Jmaiel, The temporal perspective in business process modeling: a survey and research challenges, Service Oriented Computing and Applications, vol. 9, no.1, 2015, pp 75-85.
5. Chengfei Liu, Qing Li, Xiaohui Zhao, Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue, Inf Syst Front (2009) 11, pp.201-209,DOI 10.1007/s10796-008-9089-0
6. Tung-Hsiang Chou, Jia-Lang Seng, Telecommunication e-services orchestration enabling business process management, Transactions on emerging telecommunications technologies, 2012(23), pp.646–659.
7. C. Xie, H. Cai, L. Jiang, “Ontology Combined Structural and Operational Semantics for Resource-Oriented Service Composition”, Journal of Universal Computer Science, vol. 19, pp. 1963-1985, 2013.
8. R. Alarcon and E. Wilde. RESTler: crawling RESTful services. In Proceedings of the 19th international conference on World wide web, 2010. ACM,
9. Joe Zou, Jing Mei, Yan Wang, From Representational State Transfer to Accountable State Transfer Architecture, 2010 IEEE International Conference on Web Services(ICWS2010), IEEE Press, 2010, pp.299-306, DOI 10.1109/ICWS.2010.56
10. Incheon Paik, Wuhui Chen, Michael N. Huhns, A Scalable Architecture for Automatic Service Composition, IEEE TRANSACTIONS ON SERVICES COMPUTING, vol. 7, no. 1, 2014,pp82-95.
11. Hongxia Tong, Jian Cao, Shensheng Zhang, Minglu Li, A Distributed Algorithm for Web Service Composition Based on Service Agent Model, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, vol. 22, no. 12, 2011,pp.2008-2021.
12. C. Madhavaiah, Irfan Bashir, Syed Irfan ShafiDefining, Cloud Computing in Business Perspective: A Review of Research, Vision,16(3),2012: 163–173.
13. Thomas Boillat1, Christine Legner,From On-Premise Software to Cloud Services: The Impact of Cloud Computing on Enterprise Software Vendors’ Business Models, Vol.8,Issue 3,(2013): 39-58.
14. Benedikt Martens, Frank Teuteberg, Decision-making in cloud computing environments: A cost and risk based approach, Information Systems Frontiers , Volume 14, Issue 4, September 2012, 871-893.
15. Milinda Pathirage, Srinath Perera, Indika Kumara, Sanjiva Weerawarana, A Multi-tenant Architecture for Business Process Executions, 2011 IEEE International Conference on Web Services (ICWS2011), IEEE Press, 2011, pp121-128, DOI 10.1109/ICWS.2011.99

16. J. Damasceno, F. Lins, R. Medeiros, B. Silva, A. Souza,etc, Modeling and Executing Business Processes with Annotated Security Requirements in the Cloud, 2011 IEEE International Conference on Web Services(ICWS2011), IEEE Press, 2011, pp 137-145, DOI 10.1109/ICWS.2011.78
17. Paul Hofmann, Dan Woods, Cloud Computing: The Limits of Public Clouds for Business Applications, IEEE INTERNET COMPUTING, NOVEMBER/DECEMBER 2010, pp.90-93.
18. Qing Li, Ze-yuan Wang, Wei-hua Li, Jun Li, Cheng Wang, Rui-yang Du, Applications integration in a hybrid cloud computing environment:modelling and platform, Enterprise Information Systems, 2013,Vol. 7, No. 3, 237–271, http://dx.doi.org/10.1080/17517575.2012.677479.
19. Kwang Mong Sim, Agent-Based Cloud Computing, IEEE Transactions on Services Computing, Vol.5, No.4(2012), pp.564-577.
20. Amir Vahid Dastjerdi, Rajkumar Buyya, Compatibility-Aware Cloud Service Composition under Fuzzy Preferences of Users, IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 1, JANUARY-MARCH 2014, pp.1-13.
21. Jawwad Shamsi, Muhammad Ali Khojaye, Mohammad Ali Qasmi, Data-Intensive Cloud Computing: Requirements,Expectations, Challenges, and Solutions, J Grid Computing (2013) 11:281–310, DOI 10.1007/s10723-013-9255-6.
22. Christine Mayr, Uwe Zdun, Schahram Dustdar, Enhancing traceability of persistent data access flows in process-driven SOAs, Distrib Parallel Databases (2013) 31:1–45,DOI 10.1007/s10619-012-7102-6.
23. Venugopal Srinivasmurthy, Sanjeeva Manvi, Ravi Gullapalli, et al.Web2Exchange: A Model-Based Service Transformation and Integration Environment, Proceedings of IEEE International Conference on Services Computing 2009(ICWS2009), IEEE Press, (2009): 324-331
24. Lana Abadie, Franck Di Maio, Wolf-Dieter Klotz, Kirti Mahajan, Denis Stepanov, Nadine Utzel, Anders Wallander, The self-description data configuration model, Fusion Engineering and Design, 87 (2012), pp.2213-2217.
25. R. T. Fielding, REST: Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000.
26. Federica Paganelli, Stefano Turchi, Lorenzo Bianchi, Lucia Ciofi, Maria Chiara Pettenati, Franco Pirri, and Dino Giuli, An information-centric and REST-based approach for EPC Information Services, Journal of Communications Software and Systems, Vol.9, No.1, 2013, pp.14-23.
27. Saroj Kar, Gartner: APAC Public Cloud Spending to Reach $7.4 Billion in 2015, January 23(2015), http://cloudtimes.org/2015/01/23/gartner-apac-public-cloud-spending-to-reach-7-4-billion-in-2015/
28. Qiong Wang. Three-in-one Service System on Fujian Transportation and Logistics Trading Platform, 2015-06-29,http://fjnews.fjsen.com/2015-06/29/content_16295324_all.htm (in Chinese)
29. M. Reza Rahimi, Jian Ren, Chi Harold Liu, Mobile Cloud Computing: A Survey, State of Art and Future Directions, Mobile Networks and Applications, vol.19, no. 2, 2014, pp.133-143.
30. Lina Barakat, Simon Miles, Michael Luck, Efficient adaptive QoS-based service selection, Service Oriented Computing and Applications, vol. 8, no. 4,2014, pp.261-276.