Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

ORIPS: An Open Resource-based Integrated Platform System for business process execution

Hongming Cai[1, 2]
[1]School of Software
Shanghai JiaoTong University
Shanghai, China
hmcai@sjtu.edu.cn, hongming.cai@bit-institute.com

Dominik Englert[2] and Hao Yu[2]
[2]BIT Institute
University of Mannheim
Mannheim, Germany
{dominik.englert | hao.yu}@bit-institute.com

***Abstract*—An open resource-based integrated platform system is built for the execution of business processes, assuming that the business process model is the fundamental factor for the next generation of integrated business information systems. First of all a resource meta-model is constructed, which represents the basic information unit. Then the concept of domain ontology is used to organize and manage resources obtained from distributed and heterogeneous sources. Based on the resource, Bite is used to compose RESTful services in order to control business processes in an orderly way. Therefore, the way of services to call resources within the limits of traditional web service architectures is being transferred into a resource flow. A software architecture is proposed and tested, based on an exemplary business process scenario. The result indicates a new research direction for the application of business process models based on the platform ORIPS.**

Keywords—*ORIPS, RESTful web service, resource model, ROA, business process, ontology, distributed systems*

I. INTRODUCTION

Information systems within the enterprise always involve various people, processes, data, documents, rules and other factors. In order to share information in a more effective way, we need an information system architecture based on collaboration concepts to execute business process models. Therefore, an open, flexible and in addition integrated software platform is required for successful implementations of information systems. The approach of the Service-oriented Architecture (SOA) seems to be able to solve these requirements. SOA is an abstract concept for the creation of application landscapes within and across organizational boundaries based on the composition of loosely coupled components [1]. The basic concept for the realization of SOA can be seen in the Web Service technologies, which are structured in a stack model, including SOAP, the Web Services Description Language (WSDL) and the Business Process Execution Language (BPEL) [2].

In the recent past there grew an increasing interest in the resource-oriented approach proposed by the Representational State Transfer (REST) because of the complexity of the Web Service specifications. Furthermore requires REST far fewer development steps than e.g. SOAP. The Resource-oriented Architecture (ROA) is based on the concept of the resource, which is an abstraction of information [3]. Every resource has a uniquely identifier in form of a Uniform Resource Identifier (URI), which contains the name and the address of the resource. The access to resources is being ensured through RESTful Web Services using the Hypertext Transfer Protocol (HTTP). The response from the server includes the representation of this resource. In order to support the dynamic, flexible and scalable execution of business processes, some extensions of the resource-oriented paradigm with focus on business entities or respectively resources have to be proposed [4]. But semantic relation between these resources in the business process is lack to some extent. Therefore, we are realizing an Open Resource-based Integrated Platform System (ORIPS) for the execution of business process models to reach a flexible as well as open enterprise infrastructure in order to integrate heterogeneous business applications for the collaborative work within and across enterprises. Relating to semantic relations between different resources, one of the centric components of ORIPS is the access, management and use of resources implemented on the basis of domain ontology.

II. THE MAIN IDEA OF ORIPS FRAMEWORK

Fig. 1 illustrates the realization of our RESTful Web Service-based architecture for the execution of business process models. The construction of specific mapping mechanism is stringent required to map the business process model respectively the particular process steps on the resources. Within this scope a resource can be a personnel resource, which carries out the tasks of business processes, a data resource, which is an integrated and comprehensive data source that makes data identifiable and accessible, a tool resource, which supports a personnel resource in their task execution, et cetera. The business process model generated during the build time includes the goal, tasks and rules. Furthermore it is used to control the execution of the process, i.e. to control the process thus resource and service match and carry out. The key point is that the coordination through a resource array flows in a serialization of services contrary to the traditional way of calling the resource individually through services. This represents a more efficient way for the services to call the resources in a sequence order pattern.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

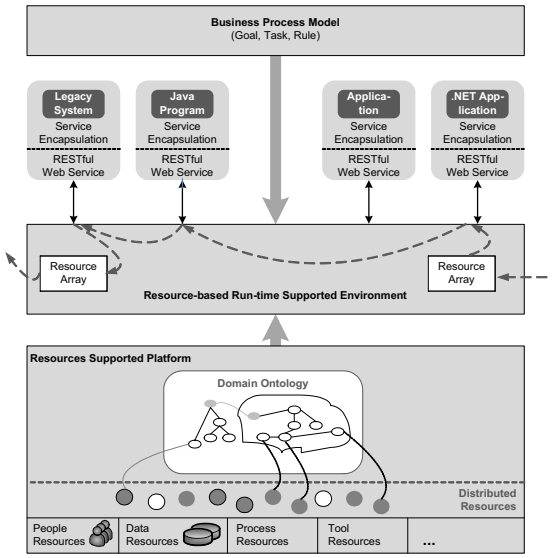

Figure 1. Concept of ORIPS for the execution of business process models

Service encapsulation is the main approach to integrate applications in enterprises in the context of SOA. Hereby the execution of the business processes takes place on a high service level instead of a low level within one system. In order to realize a flexible as well as open enterprise infrastructure, service encapsulation is used to construct RESTful Web Services for the interaction with legacy systems, java programs, application interfaces, and .net applications based on one resource oriented platform. The core of the concept for the retrieval, organization and management of the resource model and resources is the involvement of domain ontology. The creation of relations between distributed resources with possibly different representation formats and resource models occurs through mapping into one united virtual space. The domain ontology contains not only features of the resources but also operations, rules, events as well as messages. In fact we are able to build the semantic relations between resources and resource models on the basis of domain ontology and to execute complicated operations such as association, comparison, et cetera.

III. RESOURCES AND RESTFULL SERVICES MODELLING

A. Business process to ORIPS model mapping

According to DAVENPORT a business process is *"simply a structured, measured set of activities designed to produce a specified output for a particular customer or market"* [5].

In the world of SOA a process invokes external services and is itself exposed as a service. In the context of ROA, however, process interacts with resources and itself exposed as a resource. With HTTP PUT on the process, the instance of process is created and the corresponding URI is returned.

Within the scope of ROA data can be regarded as resources naturally. Data represent the business objects in the enterprise and can be modeled using e.g. the Unified Modeling Language (UML) or the Entity Relationship Model (ERM).

It is important to note, that not every business object should be considered as a resource. For example, quote and quote items should be together defined as one resource instead of two separated. So the non-resource business objects have to be firstly eliminated from the data model. We define three types of business objects ranked from the point of view of different URI representations. The first one is the so-called absolute independent business object. This type does not rely to other business objects. Accordingly it always appears in the first segment of a URI. The second one is the semi independent business object, which can be used alone as well as referenced by other business objects. Accordingly this type is able to appear in any segment of a URI. The last one is the type of dependent business objects, which cannot exist alone. The lifecycle of such business objects depends on the lifecycle of others. Correspondingly they can never appear at the first segment of a URI.

Organizational units are responsible for the execution of their dedicated activities. They can also be modeled in form of resources and their resource representation can also be organized based on the hierarchical structure of the organization units within the bounds of the enterprise. Control flows represent the path in which activities are executed. They are better described in an independent language in order to achieve the business flexibility and agility. It will be discussed later in this paper.

Activities get an input and produce a defined output. We can consider both of them as a set of data in certain states. So activity can be regarded as the transformation of data states. We distinguish between two types of activities, simple and compound activities. A simple activity operates on one main resource and possible other related resources. It can then be modeled as one RESTful service operating on the main resource using the standard interface. In contrast to this a compound activity operates on at least two groups of resources, which are not related. In that case the compound activity should be divided into a set of simple activities so that they can be modeled with some RESTful services.

B. Resource Meta-Model

To achieve the goal of comprehensive process integration, first of all the resources in an existing distributed and heterogeneous system landscape have to be identified and encapsulated. Then resources will be registered and managed within the resource integrated platform, so as to shield the differences among the different distributed, heterogeneous systems. At last, resources are able to achieve interoperability in the way of RESTful Web Services. Regarding the handling of resources, a business process step contains the procedure, in which the business information of resource models are carried out and properties of resource models are changed according to certain operations, and the corresponding parameters and messages are sent to the next business process step. Therefore, it is necessary to define a unified resource modeling method. There exist a lot of researches about the area of resource modeling, some of the studies use object-oriented modeling methods and define the structure and the relationship of resource models through mechanisms such as inheritance and encapsulation. Others

analyzed resources along multi dimensions and described resources using ontology-based concepts and approaches from the research area Semantic Web. Based on workflow analysis, a resource meta-model is proposed at this point. By this method of modeling resources, the integrated platform can organize and manage the static and dynamic messages in the enterprise system, as well as establish an open, extensible framework for business application systems so as to satisfy the integration requirements of distributed, dynamic, flexible, and typically heterogeneous systems. Fig. 2 illustrates the resource meta-model, which is accurately defined in the following.

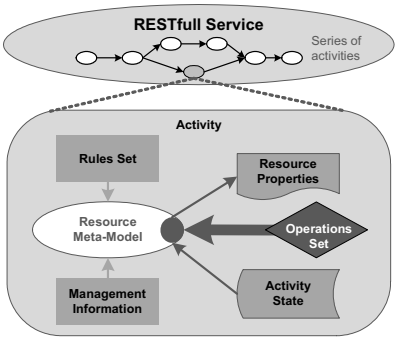

Figure 2. Resource Meta-Model

Definition 1 (Resource Meta-Model): The Resource Meta-Model (RMM) is described as a set of Resource Properties, Management Information, Operations Set, Rules Set and Resource View, and the relationship RE:

RMM = (RP, MI, OS, RS, RV, RE)

Definition 2 (Resource Property): A Resource Property is being presented by the following quadruple:

RP_i *= (ResID, ResType, SProps, DProps)*

Hereby the ResID is the unique identification of a resource entity. ResType is the underlying category, it is the description of a class of resources with a specific static property and it is able to organize the resource types by using a hierarchical tree method. SProps includes the static properties of a resource, the values of these properties will never be changed when the resource object is set up, e.g. the manufacturing date of a machine. The dynamic properties of a resource can be seen in DProps, these properties are used to describe the usage calendar, real-time status and lifecycle status of resource objects and so on, e.g. the status (created, reviewing, freezing, et cetera) of orders in a sales process. Dynamic properties can describe the status of resources in business activities and are suitable to workflow modeling.

Definition 3 (Management Information): Management Information (MI) is defined in form of the tuple:

MI = (SysInfo, PlatformInfo)

SysInfo is the information in the system where resources are stored. PlatformInfo is the register information of resources in the integrated platform. It is used to confirm the RESTful Web Service presentation method of resource objects.

Definition 4 (Operations Set): An Operation Set (OS) is defined as a set of several functions O_i with the property and status of the resource object as its parameters. It can be mapped to the specific property relationship operation. E.g. the order resource contains the operations of: create, delete, modify, freeze, activate and so on.

Definition 5 (Rules Set): A Rules Set (RS) is defined as a set of mapping functions R_i, which are able to map functions from a resource property to a specific workflow status. It is used to reflect the rules and restricts of a resource status transition in the processes. E.g., when the order's goods are availabe in the inventory, the status can be converted to *activated.* Business rules are organized based on resources, and the choreography of resource services can be more convenient through compounding resource rules in the integration of processes.

Definition 6 (Resource View): The Resource View (RV) is the result shown by the resource properties according to a certain rule. The core of the resource view is the realization of a set of rules *f (RP)*. It is used to provide the functions of resources such as display, search, filter and buffer.

RE_i is a relationship used to connect resources. After the resource-oriented analysis and reorganization of the selected system, a set of fine-granular business resources will be the resulting output. By creating virtual resources, resources are unified operated, monitored and managed. For example we can define a cluster of resources, which occur usually together in the business activities as a virtual resource, thus avoid the data interaction of a large number of fine-granular service resources, and simplify the choreography of resource services. In order to create virtual resources, the following main relationships among resource models, namely inheritance, aggregation, association, and combination can be used.

C. Ontology for the organization of resources

To take advantage of resources efficiently, resources are grouped according to similar features and preferences related to the business process model. Therefore, ontology is built, where entities, operations, and conceptional relations are included for the organization and utilization of resources. Based on ontology of a concrete domain, a pushing mechanism can be built for a resource array flow. Those are arranged in a series of services to make services interact with resources more efficiently. For further details concerning ontology see [6]. Resources of domain ontology may have complex and dynamic connections to fulfill different disposed requests. Fig. 3 illustrates a fragment related to a given quote process. For example, the resource *Quote* could link to the resource *Customer* by a request relation and the resource *Product* and *Purchase Order* are linked to get detailed information about the product price, delivered daily and so on. If more information related to the inventory and stock is needed, more Resource models such as Good Receipt, Purchase Order Request, Invoice, and Bill of Materials (BOM) will be involved. One point is that the conception in the domain ontology includes some operations inside the resources, which are references for the

organization of resources according to extern RESTful services in the run time of a business process model.

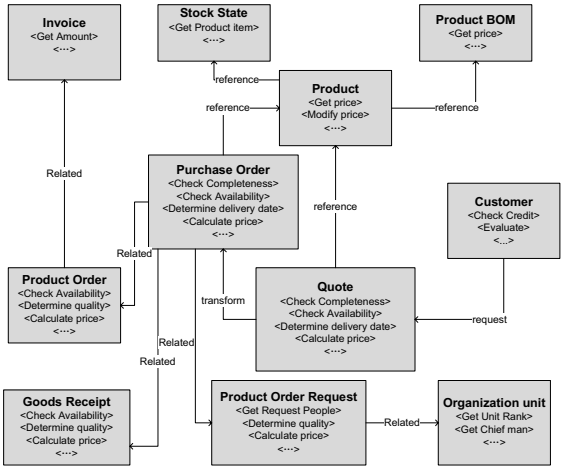

Figure 3. Fragment of the domain ontology related to the quote process

Based on this conception of certain domains, some complex operations between two resources such as similarity calculation and associated navigation could be executed. This facilitates the building of semantic relations between resources as well as to use these resources in a deeper level.

D. Representation of Resources and RESTful services

In the context of ROA resources are represented through URI, as mentioned. The uniform URI template for resources consists of three parts:

"/{namespace}/{name}?{parameters}"

Hereby namespace is used to avoid possible name conflicts in the heterogeneous environment because so many resources exist. Furthermore based on namespaces the resources can be better structured. For example, to represent organization units we can use a namespace to determine the organizational structure that the organization unit belongs to, such as */SalesDepartment/Regional/*. Another example is the dependent business object, whose namespace contains the business object it relies on, such as */customer/quote*. A name is used to identify a resource. Because of the variant names of the same resource ontology is used to avoid such kind of ambiguity. Parameters are used to help defining the limits of resources. For example, an identifier of a resource is used as a parameter to get a certain resource. Resources can be represented in different forms. In the context of HTTP they are MIME media types. Typical types are, e.g., *text/html* for web pages, or *application/json* for JavaScript Object Notation. RESTful services use standard HTTP methods (GET, PUT, POST, and DELETE) to access or modify the resources. Given the resource URI, the GET method is used to retrieve the particular state of resources. With PUT a new resource, which is represented in form of the given URI, will be created, which can be deleted later with the DELETE method. POST changes the resource state, which, together with parameter, is most flexible to support possible actions on resources. Parameters are used to designate, what kind of state change should be made on the resources. For example, to check the availability of a quote, the method POST will be issued to the URI */customer/quote?ac=available*. Based on this way of representation business processes can be fully supported in the context of Resource-oriented Architecture. To compose the RESTful services we use the Bite [7] language, which is a minimalist choreography language and runtime built to support the Web. Bite is similar to BPEL but more lightweight. It contains particularly two main constructs: activities and links. An activity defines a unit of tasks and links define dependencies between activities. The basic constructs in Bite are the following: <receive>, <reply>, <receive-reply>, <invoke>, <local>, <wait>, <assign>, <pick>, <while>, and <source>. Within the scope of Bite data-centric flows as well as interactive flows are supported. For further details of Bite see [7].

IV. Case Study And Discussion

A. Architecture of ORIPS

Based on resources and RESTful web services, Fig. 4 illustrates the framework of ORIPS, which is divided into six components explained in the following:

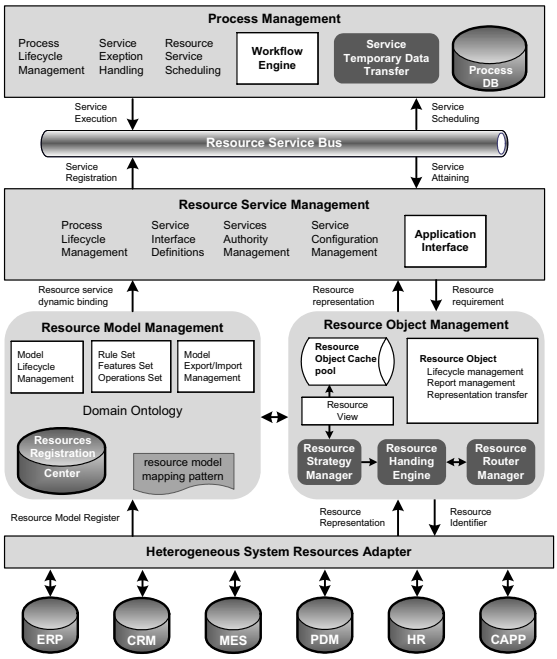

Figure 4. Framework of ORIPS

(1) Heterogeneous System Resources Adapter: A Resource Adapter supports heterogeneous systems to encapsulate operational resources, and then registering them to the Resources Registration Center on the integrated platform in order to make sure that the integrated platform can use the unified resource identifier to implement operations on the heterogeneous resources.

(2) Resource Model Management Module: This module is responsible for the management of the resource model information from various heterogeneous information systems. Domain Ontology is an essential reference to organize the

resource models. The meta-information of the resource model and resource mapping strategies are stored in XML files and can be used by the resource object model. This module also provides the service dynamic binding model information to the Resource Service Module.

(3) Resource Object Management Module: This component manages the resource object information, which is obtained from different heterogeneous systems. The resource request from the Resource Service Module will be assigned through the Resource Strategy Manager to the corresponding Resource Handling Engine, and then the parsed request will be routed by the Resource Routing Manager and integrated with the Resource Presentation from the heterogeneous systems and sent back to the Resource Service Module. URI is used to integrate resources to the Resource Object Cache Pool and enhance the performance of the integration process.

(4) Resource Service Management Module: The lifecycle, discovery and matching, transaction and security of resource services are managed within the limits of this module. It receives service dynamic binding requests and after the service interface definition and configuration it publishes them in the format of a Web Application Description Language (WADL) document [8], so that the external applications can call these services.

(5) Resource Service Bus: It ensures the information transfer and the routing among resource services. It includes the publishing, subscription, response to request of resource services, and it supports the synchronous or asynchronous messages, as well as records the call history of services, measures and monitors data.

(6) Process Management Module: This component is responsible for the management of process service lifecycle and choreography, and it provides the workflow engine to parse the resource process execution language. Service Temporary Data Transfer Module embeds the data handling scripts into the resource service process files, and executes the operations extraction, filtering, cleaning, and transfer on the input and output parameters among services.

B. An exemplary case scenario

The following example, depicted in Fig. 5, demonstrates how to use our approach to implement a business process in the context of ROA. A customer sends a request for a quote (RFQ) to the sales department of a company. In the first instance the sales employee checks the completeness of the request. If some data lack, the customer will be requested to complement the request. In the second step the type and quantity of products that are requested are checked for their availability in the stock. If they are not available, the request will be rejected. In the case of availability the delivery date is determined and the price is calculated. A quote will then be sent to the customer.

First of all we analyze the resources involved in the process. Based on our approach the quote items are part of the quote and should not be treated as resources. The quote, however, is a resource which contains the quote items as well as the quote. Although there is a request for a quote in the process, we think it is just a status of quote. Because a quote relies on the customer, it is classified as a dependent resource. A possible URI for the quote is then */customer/quote*. Hereby the customer is classified as an absolute independent resource and can be represented with the URI */customer*. A Product can be independent as well as dependent, which is referenced by a quote item. Accordingly a possible URI for the product is then */product* or */customer/quote/product*. The mentioned resources and their relationships are illustrated in Fig. 3. There exist mainly five activities in the activity diagram that relate to the resource handling (Fig. 5). They are all simple activities because they operate on one resource, namely the quote. At first a PUT method is issued on the resource */customer/quote* to create a new quote with initial status. For the activity *check request for quote* a POST method is issued on the resource */customer/quote?ac=checkCompleteness*. Hereby a parameter is used to explicitly identify the change of the status. All possible operations on a resource are described within ontology, accordingly there exists no ambiguity regarding to the semantics of this operation. The activities *check availability*, *determine delivery date* and *calculate price* can be handled similarly. To get the status of the quote a GET method can be issued on the resource */customer/quote*.

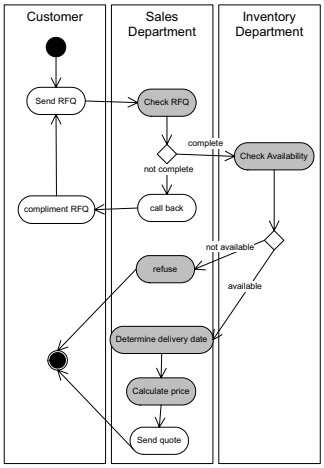

Figure 5. Example scenario quote process

In the Table 1, the whole process is designed in the Bite-language.

TABLE 1. PROCESS EXPRESSED WITH BITE

```
<process name="requestForQuoteProcess">
        <receive name="requestForQuote"
                url=http://example.com/requestForQuoteProcess />
        <invoke name="checkCompleteness"
                invocationTarget="http://example.com/customer/
                        quote?ac=checkCompleteness">
                <source name="requestForQuote" input="yes"/>
                <input value="requestForQuote" />
        </invoke>
        <reply name="requestForCompleteness">
                <source name="checkCompleteness"
                        condition="checkCompleteness==no" />
                <input value="Please complete the request for quote
            with the url  http://exmaple.com/customer/quote" />
        </reply>
```

```
        <invoke name="checkAvailability"
                invocationTarget="http://example.com/customer/
                        quote?ac=checkAvailability">
                    <source name="checkCompleteness"
                            condition="checkCompleteness==yes"
                            input="yes"/>
                    <input value="requestForQuote" />
        </invoke>
        <reply name="refusal">
                    <source name="checkAvailability"
                            condition="checkAvailability==no" />
                    <input value="The products you requested are not
                                available." />
        </reply>
        <invoke name="determineDeliveryDate"
                invocationTarget="http://example.com/customer/
                            quote?ac=deterDeliveryDate">
                    <source name="checkAvailability"
                            condition="checkAvailability==yes"/>
                    <input value="requestForQuote" />
        </invoke>
        <invoke name="calcPrice"
                invocationTarget="http://example.com/customer/
                                quote?ac=calcPrice">
                    <source name="determineDeliveryDate"/>
                    <input value="requestForQuote" />
        </invoke>
        <reply name="quote">
                    <source name="calcPrice"/>
        </reply>
</process>
```

C. Discussion

Compared with other approaches, which use the abstract concept of resource-oriented architecture to support business processes [4, 9], our approach is highlighted by the following features:

- All basic elements of a business process are mapped to corresponding elements in resource-oriented architecture to build a basis for the implementation. Unlike the mapping of tasks on resources in [9], we propose the corresponding mapping of RESTful services on resources, which is natural. Compared with [4], which proposed the information-centric business modeling and which requires the transformation of usual task-centric business modeling, our approach does not require suchlike transformation.
- Unlike the use of micro-formats to define the next-step actions in [9], which burdens the client by transferring the control of a business process to the client, Bite is used to compose RESTful services so that a central control of a business process on the server-side is possible.
- One of the main problems within the scope of ROA is the production of a large number of objects [10]. With our REST-based approach this problem can be partly avoided. On the one hand, a resource is assigned to a well structured URI, so the resource management can be executed in a much more efficient way. On the other hand, ontology is introduced to build semantic relationships between resources.
- Furthermore a resource meta-model provides a unified information framework for the formal description of resources. By introducing the elements process, activity, people, data and management information into one body, the resource has much more semantics than other general formats such as UML, ERM and so on.

V. CONCLUSIONS

Based on RESTful web services, a resource-based integrated platform for the execution of business processes is proposed. Comparing to SOA and other ROA platforms, ORIPS focuses on semantic relations among the resources. Based on the concept of domain ontology, a resource in the platform is organized effectively in order to support the matching of services and resources. Furthermore an exemplary business process is given for testing the platform. The further research will focus on the mechanism of pushing ontology-based resources to a serial of services.

ACKNOWLEDGMENT

This paper is supported by the National High Technology Research and Development Program of China (“863” Program) under No.2008AA04Z126, the National Natural Science Foundation of China under Grant No.60603080, No.70871078, and the Aviation Science Fund of China under Grant No.2007ZG57012. Furthermore we want to express our gratitude to the Alfried Krupp von Bohlen und Halbach Foundation for supporting this joint research.

REFERENCES

[1] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Upper Saddle River: Prentice Hall, 2005

[2] S. S. Kumar, and S. Sinha, “Limitations of Web Service Security on SOAP Messages in a Document Production Workflow Environment,” 16th International Conference on Advanced Computing and Communications, 2008, IEEE Press, pp. 342 - 346

[3] L. Richardson, and S. Ruby, RESTful Web Services, Cambridge: O’Reilly, 2007

[4] S. Kumaran, R. Liu, P. Dhoolia, T. Heath, P. Nandi, and F. Pinel, “A RESTful Architecture for Service-Oriented Business Process Execution,” Proceedings of IEEE International Conference on e-Business Engineering 2008, pp. 197-204

[5] T. H. Davenport, Process Innovation: Reengineering Work through Information Technology, Boston: Harvard Business School, 1992

[6] R. Studer, S. Grimm, and A. Abecker, Semantic Web Services: Concepts, Technologies, and Applications, Heidelberg: Springer, 2007

[7] F. Curbera, M. Duftler, R. Khalaf, and D. Lovell, “Bite: Workflow Composition for the Web,” Proceedings of IEEE International Conference on Service Oriented Computing 2007, pp. 94-106

[8] T. Takase, S. Makino, S. Kawanaka, K. Ueno, C. Ferris, and A. Ryman, Definition Languages for RESTful Web Services: WADL vs. WSDL 2.0, IBM Reasearch, 2008

[9] X. Xu, L. Zhu, Y. Liu, M. Staples, “Resource-Oriented Architecture for Business Processes,” 15th Asia-Pacific Software Engineering Conference 2008, pp. 395-402

[10] M. Muehlen, J. V. Nickerson, and K.D. Swenson, “Developing web services choreography standards: the case of REST vs. SOAP,“ in: Decision Support Systems, vol. 40(1), 2005, pp. 9-29