International Journal of Multimedia and Ubiquitous Engineering
Vol. 10, No. 2 (2015), pp. 137-152
http://dx.doi.org/10.14257/ijmue.2015.10.2.13

Research of the Interconnection of Workflow System Based on Web Service

Gang Yuan[1], Rui-zhi Sun[1], Yong Xiang[2] and Yin-xue Shi[1]

[1]Key laboratory of Agricultural Information Acquisition Technology (Beijing), Ministry of Agriculture P. R. China, China Agricultural University, Beijing 100083
[2]Department of Computer Science and Technology, Tsinghua University, Beijing 100084
yuan7_7@163.com

Abstract

In order to achieve the interconnection between different workflow management systems, it was proposed that all the distributed workflow systems would be encapsulated as web services to perform the entire business process collaboratively by the way of processes' composition in this paper. By analyzing the comparison between the composition of processes and ordinary Web service, we studied interactive control, the parameters required to be passed through the distributed workflow systems, the workflow system service's interfaces and its packaging. Furthermore we put forward a general method of the workflow systems interactive interfaces' extension and the way of the workflow service's encapsulating and invoking. By this approach, it can easily combine the processes or process fragments which deployed on different workflow systems without other agents and components. It also provides support for the interconnection of the workflow systems in distributed environment, and ultimately achieves a coordinated operation between different workflow engines.

Keywords: *Workflow System, Interconnection, Web Service, Interface*

1. Introduction

Workflow management systems have been widely used in all kinds of enterprises for their ability to describe and execute business processes in a way that make it easy to coordinate information and resources among person or groups as a sequence of operations. At present, there exist a variety of workflow products that vary in workflow model, process description language and the system's functions [1]. On the other hand, the business processes of modern enterprises gradually reveal cross-sectoral, cross-regional and even across different companies. In order to better support business process reengineering and improve the performance bottleneck of centralized workflow engine, we can increase or decrease the number of execution engine, personalize and optimize the performance of engine and define the business activities flexibly according to businesses' needs. Therefore, a business process' implementation will be interacted and collaborated between different workflow management systems of different departments or companies, it's essential to execute the processes collaboratively and interactively between these systems and tools in a heterogeneous environment.

ISSN: 1975-0080 IJMUE

2. Related Work

The combination of cross-system workflow business processes has been fully considered while the workflow management system was proposed. The Workflow Management Coalition (WFMC) defined the workflow engine could communicate with others collaboratively, and the interface 4 of its reference model is used for different workflow systems or engines' interaction [2]. WFMC also proposed interconnection must have two main aspects: (1) the public explanation of process definition needs to be extended (2). It should support the conversion and transmission of different types of control information, workflow process data and application data between different execution servers. It mainly used the conversion of workflow language to unify the description of workflow process definition. Fernando [4] and Plankensteiner [12] mapped part or the whole process to IWRL which is a kind of formal intermediary language to realize the interoperability between different workflow systems. In the long run, considering the languages' transmission for processes' interaction during processes modeling will facilitate the combination of processes, the users can combine various existing processes and modify the process model at any time before the execution of the process. But there do not have a fully standardized workflow language that can be completely mapped with existing workflow languages.

The workflow languages could not be completely standardized or unified, therefore workflow engines' interaction can be seen as a key way to achieve different workflow systems' interconnection. With the concepts' put forward and technologies' realization of SOA, grid and middleware, business processes' integration and workflow systems' interconnection based on Web Service become the main technical means. Jiang [10] set each enterprise as a child node of P2P network by a distributed workflow management system which based on P2P architecture, and each node could perform a process instance collaboratively with other nodes through JXTA network interface. Xu [11] used EAI technology to build dynamic interoperability between various heterogeneous processes based on subprocess model, multi-layer dynamic state transition model, communication and automatic monitoring components, but this method needs external components and channels. Pavlin [5] established communication among different packaged workflow services through each heterogeneous workflow system's domain knowledge, the main consideration of services' integration and heterogeneous data's transmission was based on ontology. Zhen [13] proposed a Message-Oriented Middleware integration and SOA-based approach to solve the integration of business data and processes.

During the process' operation, a variety of information transfer between process services, including control information, workflow data and application data and so on. The caller of the service will do some response and handling according to the received information, and then realize the processes' interaction. Therefore, the parameters' type and category also need to be fully taken into account. Kukal [6] and Korkhov [9] established a link between each node of workflow engine service in grid system by considering the processes' invocation mode and the way that the users could awake and invoke workflow systems which act as a web service. But this implementation in grid system also needs to rely on some components and agents. Alqaoud [7] built the interoperability during the workflow processes' operation by notification message system based on web service, the main method is using publish-subscribe mechanism to publish a workflow system which is in use for other workflow systems to subscribe and receive the returned message in the implementation of an event.

In introducing their own methods of the research that mentioned above, they also need to rely on the third platform or components to realize the interaction between heterogeneous processes. In order to use a general method and other components would be used as little as possible, Yang [3] proposed to package each sub workflow system which is belong to a

distributed workflow management system into web service and provide collaborative workflow model to other sub workflow systems. Our previous work [8] realized the interconnection between XPDL and BPEL by packaging the workflow engine's start and running script into a service, but this method does not properly realize the parameters' transferring between internal and external of the workflow system.

The important aspect of workflow's interconnection during the operation is to call and wake up the process instances which may be locate in other workflow systems. If we'll integrate the workflows together which work in different environments, the communication between the client API and workflow execution service need to be standardized and unified, that means all the workflow products should have a similar interface and the sharing information between these interactive processes should be marked and classified clearly. Korkhov [9] classified the workflow engine's interfaces and packaged them into web services for clients to invoke. It can't fully meet the demand of data's transferring and engines' interaction in distributed network environment if we only use workflow engine's original interfaces, so part of the interfaces' function should be expanded.

In order to integrate the process applications into the existing workflow systems and make the workflow systems' function could be invoked by others, we also use services' combination to achieve distributed execution of business processes in this paper. We have analyzed the parameters that needed to transmit between the interconnection systems. We also studied the workflow interactive interfaces' unification, the difference between workflow system service's encapsulation and invocation from general web service and so on. In this paper, it can easily combine the processes or process fragments which deployed on different workflow systems without the aid of other agents and components, ultimately realize the coordinated operations between different workflow engines.

3. Interconnection of Workflow System Based on Web Service

In order to achieve the full implementation of a process, we can combine the services which were packaged from processes or sub-processes. The implementation is similar to the service's combination, but the manner that workflow service's invoking and combination is somewhat different with general service, mainly in: (1) This service is corresponding to sub-process or process fragments which should be performed by its own workflow engine. So we should consider how to start an off-site workflow engine and execute a process fragment (2). Since there is a close relationship between the data before and after the events in a workflow process, some parameters which actually pass between different workflow systems need to be introduced into the process fragment which we have invoked (3). The sub-processes or activities of a workflow process always execute in parallel, if the fragments were executed by different engines, the results used to return and the service requester waiting for a response need to be considered by the synchronous or asynchronous control strategy at the end of the process' completion (4). In order to achieve the real-time optimal combination of process fragments, it also need to consider the discovery and management of the processes which located in different workflow systems that we can catch the right processes which are available in the runtime. This paper only research on the first three cases.

3.1. Workflow Service Interface's Definition and Function

When a total process needs multiple distributed workflow systems to work collaboratively, the encapsulated process of each workflow system will act as one of the total process' activities. So the total process' activities may run in different workflow engines, and the

original simple data flow between the process' activities would turn into complex data transformation between different workflow systems.

The interactive data between workflow systems include process' functional identification, workflow relevant data, the mode of process invocation, the result and state of a process' completion and so on. The process' functional identification will notify the current workflow engine to start and execute which process. Workflow relevant data is the data which needed to introduce into the executed process after the implementation of its former activity, including process-related data and some business data. Invocation mode is used to represent which mode the total process needs to be executed, synchronous or asynchronous mode. The result means the result of the implementation of the process should be returned back to the total process according to the client's requirements. The completion status refers to the end of the process' execution, which is normal or abnormal.

Since it is convenient for web service to exchange data and communicate between different systems, this paper provided the functions of workflow system for client to invoke by encapsulating the workflow system's interfaces. The main function of workflow service includes the following two aspects: (1) it provides the necessary interface for external applications, such as process definition's modeling and deployment, process instances' creation and start, and the historical data's query and so on. (2) It provides data and services during the execution of the workflow engines when it need. Therefore, the packaged workflow service's interfaces are designed as follows:

Definition 1: The Definition of Workflow Service's Interface

Definition 1.1: （Workflow Service Input Parameters）WSIP=<pf, ts, bda>

（1）"pf" indicates process definition's information which demonstrates the functional factors of the process, and **pf=<pid, pn, op, pd>**. Among them, "pid" is identifier of process definition, "pn" is process' name, "op" is process' operator and "pd" is process' description.

（2）"ts" is the mode of service invocation, and **ts=<syn OR asyn>**. Among them, "syn" indicates synchronous mode and "asyn" indicates asynchronous mode.

（3）"bda" is the relevant business data which introduced from the total process.

Definition 1.2: （Workflow service output Parameters）WSOP=<ts, pr, es>

（1）"ts" is the mode of service invocation, including synchronous and asynchronous mode.

（2）"pr" is the result of process, and **pr={pda, bda}, pda=<psid, psn, op>**. Among them, "pda" is the data of process instances, including process instance's ID, process instance's name and process operator; "bda" is the business data.

（3）"es" is completion status of the process, and **es={ending, suspending, exception}**, which indicates the processes' normal completion, pausing, suspending and abnormal termination. We only did some research and experiment under the situation of normal completion in this paper.

3.2. Design and Character's Analysis of the Interactive Interface

WFMC [2] has defined other workflow system's interfaces besides supporting for collaborative work between different workflow systems, just as process definition's import and export interface, workflow client interaction interface, workflow management and monitoring interface, and invoked application interface.

It should use the same application data exchanged methods or gateway mechanisms to map the data between different workflow products when they are needed to work in coordination. Workflow applications or data exchanged methods are handled by three interfaces [2] which

are client interaction interface, invoked application interface and workflow collaboration interface. Therefore, workflow service would interact with other workflow systems through these three interfaces while the process is running. The workflow engine firstly call the client application interface to decide the processes and activities' implementation according to the work items' completion and the users' choice, then the engine will call internal application interface and transfer the process into the appropriate steps based on the parameters of previous decisions. In a distributed workflow system, a total process is divided into activities and events, and its implementation will be promoted by several workflow engines' working together and depend on the transfer of appropriate resource and parameters' information between different engines. Because of the difference of the platforms, process standards, the data layer and communication protocols between different workflows, there exit a great difference in their internal and external interfaces and the workflow interfaces' extension and encapsulation should be carried out in a unified way. The design of workflow interfaces' extension and process services' invocation that proposed in this paper is shown in Figure 1.

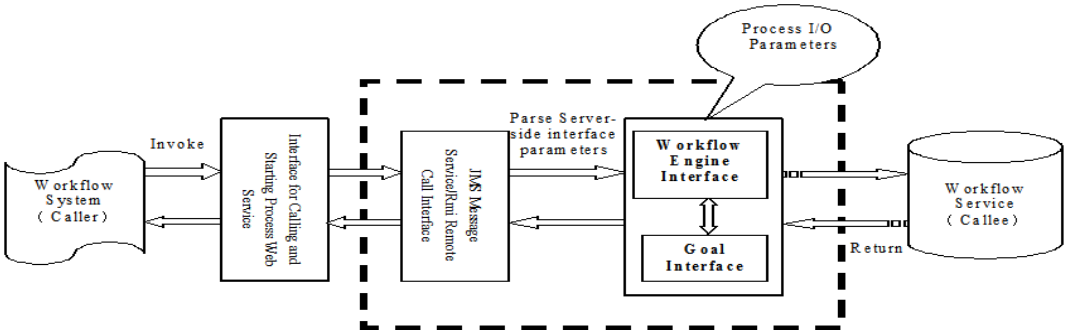


Figure 1. Extension of Process Service's Interactive Interface

Workflow relevant data could be conversed and transferred through workflow's read/write API, and also be stored in a shared object for delivering. The extended function of workflow system interface that we have proposed is mainly used for receiving the parameters passed from external workflow systems and handling the returned result. The workflow engine needs to identify the incoming data's semantics when external data passed into its system, such as the data's type, content and so on. It also needs to identify if the data is process relevant data or business data, the data which needed to be passed in corresponding process activities and which part of the data needed to be extracted to return back are according to users' demands.

Two parts of the workflow system interfaces were extended in this paper, one is the functional expansion based on the original workflow engine interfaces, and the other is adding a goal interface in which the global variables were set throughout the entire process' execution that used to obtain the incoming and outgoing data.

3.2.1. The Extension of the Workflow System Interface: The operating mechanism of traditional workflow engine mainly includes process' creation, starting, completion and the messages' sending and receiving and so on. The expansion of original workflow engine interfaces' functions include the following aspects:

(1) Create a process instance: The workflow engine generates the process instance's information (such as process' ID, name, operator and global variables) by the process' definition, and the process instance is initialized. Then the state of the instance changes to running after inserting its information into database and loading into memory. Before the start of the instance, it introduces the above parameters and starts the first activity.

At the beginning of the start activity, if the mode of invocation is synchronous, it will wake up a thread after the process has finished.

(2) Start and run a process instance: It will start and run the appropriate process according to the process instance's ID, parse the needed parameters for the running activity instances from the global variables and deal with the work items.

(3) Complete a process instance: When the process runs to the end activity, firstly the engine determines whether all the activities of the process are completed, and if it is a completion of the process. Secondly it will determine whether there exists a father process of the current process, if there has none, it will return corresponding parameters to the goal interface at the end of end activity.

Definition 2: The Definition of Workflow Engine Interface's Extended Functions

interface-wfEngine=<enf, params, returnType, codeDef, description>

（1）"enf" is original engine interface whose function are extended, and **enf={create, start, complete}**, including the interface of process' creation, starting, execution and completion.

（2）"params" is parameters of the interface, and **params={pf, pda, gp}**, **pf=<pid, pn, op, pd>**, **pda=<psid, psn, op>**.

Among them, "pf" indicates the parameters which are related to the process definition, including process definition's ID, process' name, process' operator and description of the process; "pda" indicates the data of process instance, including process instance's ID, process instance's name, process' operator; "gp" is the data which is assembled by process global variables in a specific format, and process global variables including global variables of the process definition and parameters that passed between the workflow systems. Process data between different workflow systems would pass through the "gp" parameter.

（3）"returnType" is the type of interface's return value.

（4）"codeDef" is the implementation of the interface.

```
1)  interface create(pf, gp){
        pda← created by pf    //generate process instance based on the process definition
        //load process instance's information and global variables to the database and memory
        database,memory←(pda, gp)
        status←running        //set the process instance with a status of running
        if ts=syn    //awoke thread at the completion of the process if it is synchronous call mode
          while status!=finished
              Thread.sleep(INTERVAL)
         return pda   }
2)  interface start(pda, gp){
        //parse and deliver goal global variables into activities
        activities.getbyprocessInsId(psid)←Goal.parse(gp)
        handle workitems       //handle work items }
3)  interface complete(pda, gp){
     if activity.type=end && all activities finish  //run to end activity and all activities are completed
       if currentProcessIns.hasParentProcess=false  //current completed process has no parent process
              status←finished
                //assemble the process' result according to a certain format through goal interface
                gp←Goal.generate(pr)
          return gp   }
```

（5）"description" is description of the interface.

External workflow system will call and start a workflow process via "create" and "start" interface. The process instance is executed by the invoked workflow system's engine which

determines whether the process instances is performed completely and then return the result via "complete" interface. After the workflow engine interfaces' extension, the process' main execution is shown in Figure 2.

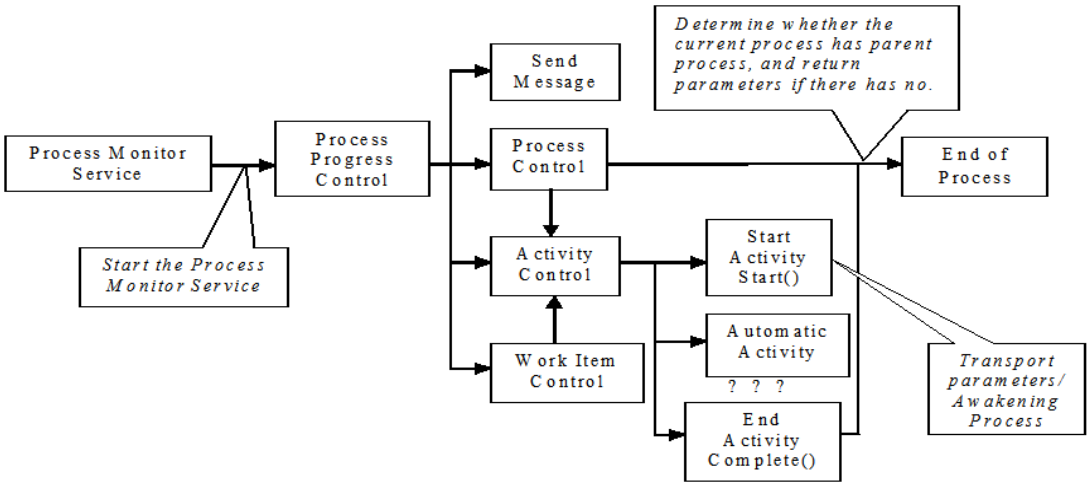


Figure 2. Execution of Interfaces-extended Workflow Engine

3.2.2. Unified Workflow Service Interface: Workflow collaboration interface [2] is similar to client application interface, it supports the exchange of different application data in different systems, and it also has the function of gateway. It relies on a specific collaborative workflow model from which an activity can be mapped as another activity or a new process/sub-process from workflow system A to system B, and establish a session between different workflow systems, so it can do some operation on workflow definition and its objects, process control, process status, event management and data handling.

In order to pass all kinds of external parameters into current workflow system and facilitate different users have the access to the results they concern, in addition to the extension of original interfaces' function, a goal interface is introduced between workflow engine and external applications. The main function of the goal interface is to receive and parse the client's parameters, create a new link to the workflow engine service, receive and interpret the response of workflow engine service and then return the result back to client. The specific approach is to establish parameters' mapping between the calling and called process, unify workflow systems' requirements for different data formats, and assemble the required external data and parameters into a xml file which acts as one of the goal interface's global variables. Firstly, the xml file would be parsed before a process' execution, secondly, it will analysis and acquire the process data which needed by the follow-up activities. Finally, at the completion of the process, the operating results and other parameters will be assembled according to a uniform format for the caller to parse and then get the appropriate data.

When the total process runs to the step that needs to call other workflow system's process, it would go through the following steps:

(1) The process data and business data of the total process would be stored in this process instance's global variables, and a relation would be established between calling and called process by mapping the input and output parameters' ID of goal interface with the bound process global variables' ID which have the same name.

(2) The parameters which needed to introduce into the called process would be assembled into an xml file and stored into goal's global variables. The xml file would be transferred to the called process' global variables through the relation of goal's reference parameters' ID

with the called process global variables' ID, and then the called process would be started and executed, the parameters would be parsed and resolved when required in follow-up activities.

(3) After the completion of the called process, the results which including process data and business data would be assembled into an xml file according to a certain format and stored into the process instance's global variables, and then the xml which will be obtained and parsed by the service invocator would be returned back to goal interface by the mapping of goal reference parameters with this process instance.

In order to match the input and output parameters of the workflow web service's interface effectively, as a reference of Definition 1 which mentioned in the previous chapter, we have defined the goal interface and designed goal interface parameters' schema as follows:

Definition 3: The Definition of Goal Interface

interface-Goal=<goalfunc, params, returnType, codeDef, description>

(1) "goalfunc" indicates the function of goal interface, and **goalfunc=<generate, parse>.** Among them, "generate" is used for assembling goal parameters into an xml file which follows the defined schema; "parse" is used for parsing xml file which is assembled by goal parameters.

(2) "params" is the input parameters of the function, and **params={goalPara, xml-file, node-name}**. Among them, "goalPara" is the goal's parameters, including the global variables of process definition, the variables which refer to result of process, and the parameters which introduced from external workflow systems. "goalPara.name" is name of goal's parameters while "goalPara.value" is the value of it; "xml-file" is the xml file which assembled by the process' "goalPara" according to the schema, and then transferred to "gp" of the engine's interfaces; "node-name" is the node name of the xml file that used for parsing some certain kind parameter.

(3) "returnType" is the type of interface's return value.

(4) "codeDef" is the implementation of the interface.

```
interface Goal{
1) generate(goalPara){    //assemble the data into a certain formatted xml file based on schema
     gp←generate xml-file based Schema(goalPara.name, goalPara.value)
     return gp
}
2) parse(xml-file, node-name){    //parse the xml file to get the desired process' result
     pr←parse xml-file by node-name
     return pr
}
```

(5) "description" is description of the interface.

Goal Interface Parameter XML Schema：

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://test.com"  xmlns="http://test.com">

<xsd:element name="ProcessFunc" type="xsd:string"/>   <!-- Process Function Parameters -->
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="parameter" type="xsd:string"/>

<xsd:simpleType name="InvokeStyle">    <!-- the Mode of Service Invocation-->
          <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Synchronous"/>      <!-- Synchronous Mode -->
                    <xsd:enumeration value="Asynchronous"/>     <!-- Asynchronous Mode -->
          </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ProcessGoal">        <!-- Goal Parameters -->
  <xsd:sequence>
                    <xsd:element ref="name"/>
                    <xsd:element ref="parameter" minOccurs="0" maxOccurs="unbound"/>
                    <xsd:element ref="result" minOccurs="0" maxOccurs="unbound"/>
  </xsd:sequence>
 </xsd:complexType>

</xsd:schema>
```

3.3. Workflow Service's Function and Encapsulation

Since different process instances may correspond to different process definitions, the number and type of workflow services' input/output parameters are not fixed, and the strong dependence on environmental resources of process' implementation, the main functions of workflow service are as follows: (1) The workflow service should receive the incoming parameters from external workflow system, and pass the corresponding process and business' information to workflow database through the internal database service. (2) The workflow service should return the results and state of the subscription process after its completion. In order to achieve these two functions, the service's access address will be bound with available application interfaces, and the collection of its ports is defined as a workflow execution service.

The design of workflow service's packaging are as follows: (1) The service should provide the necessary interfaces for external applications, such as the process definitions' deployment, process instances' start and trigger, the historical data's query and so on. (2) The workflow service should provide other required services during the engine's scheduling process, such as database service and so on. (3) The service could set the global variable schema based on the decomposition of user's demand for process services. (4) The extended interfaces are maintained consistency and opened to the users together with the original interfaces, and it won't affect the client's direct access to the original interfaces' function.

The method of workflow service's encapsulation is shown as follows:

```
public class getSynflowResult {
 step 1  //define workflow system service's common object
      WFService_Obj←{ MONITOR_RMI_ADDRESS,  //wf system's  rmi remote call
interface
                          op,                                    //process operator
                          RmiConnection,            //database rmi connection object
                          ProcessMonitor,           //process monitor object
                          psid,                        //process instance's ID
                          pr, ……            //process instance's result
                    }
 step 2  //package the workflow system service
        // input parameters: identification of synchronous or asynchronous, goal name and
parameters
   public String ReceiveSynflowResult (String Sflag,String GoalName,String goalPara){
         getSynflowResult WfObj=new getSynflowResult();
   step 2.1  //get the process data from an external workflow system
          OuterDataArray ← (goalPara.name, goalPara.value)   //goal parameters
          gp ← WfObj.generate(OuterDataArray);       //assemble the external parameters
     step 2.2  //obtain the process definition and create a process instance
       //obtain workflow service's rmi remote process monitor object
       pm ← MonitorConnection.getProcessMonitor(MONITOR_RMI_ADDRESS);
       //return the process instance's information
      s[] ← pm.create (PROCESS_NO, PROCESS_XML, "Test", op, gp);
       psid ← Integer.parseInt(s[0]);       //get the current process instance's ID
     step 2.3  //start and execute a process instance
       pm.start (psid, op, gp);
     step 2.4  //waiting for the return of the workflow service's result
              //the flag of determining whether the process is completed and return the result
              flag ← unfinished ;
                   while (!flag) {
                  pservice ← PropertyService.getPropertyService();
        rmiConnectName ← pservice.getProperty("dbrmiconnect");
                  //call the database service, establish a connection with the business database
                  conn ← RmiConnection(rmiConnectName);
          //parse the xml returned from  "complete" interface of the engine and get the
```

The parameters of the service's entrance are as follows: "Sflag" indicates invocation mode, "GoalName" is the goal's name, and "goalPara" indicates parameters of the goal. For the non-service workflow system's packaging, firstly we use Java to package the functional modules to meet JNI(Java Native Interface) calling specifications and develop the extend functions, secondly we add wrapping code through the development of local Java program, finally it will be packaged and deployed by Java as a standard Web Service.

4. Case Study

In this section, we will test the method which is described in the previous section through calling a process that exists in some workflow service by scientific workflow Kepler and

open source workflow engine ODE. Firstly, we packaged this workflow system's interfaces for external access and extended goal interface into web service through Axis2 and other components of the Web Service container. And then, we packaged and deployed non-functional components of web services automatically as a standard service into Axis2 through the Ant tool's commands which provided by Apache.

There exists a Product Acceptance Process which is actually running in a certain workflow system. The product acceptance process is used to describe the leader issued the product detection forms to several departments, and then the forms would be distributed to some staffs and reviewed by the leader at last. The whole process is shown in Figure 4(a), M and N represent the number of department and person for distributed separately.

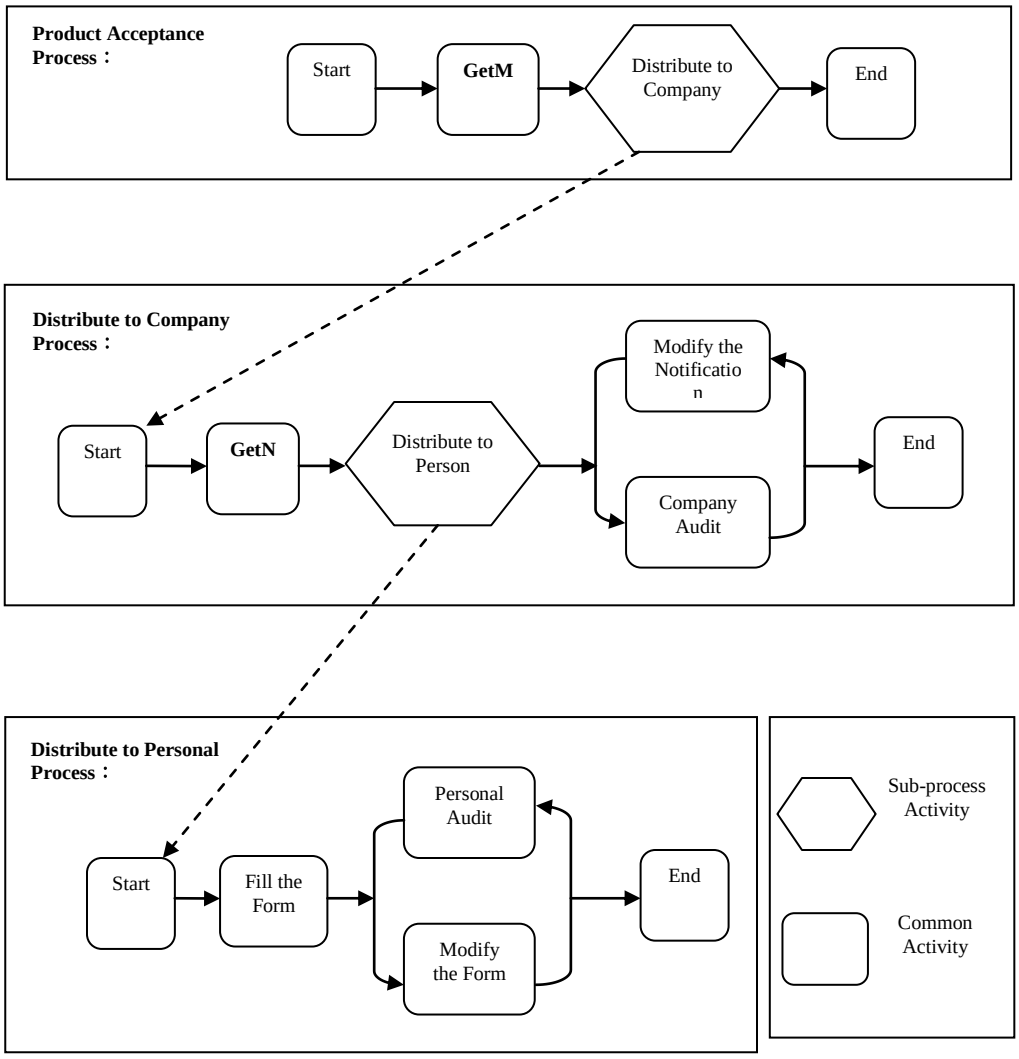


Figure 4(a). Product Acceptance Process

A process can be packaged into a service which acts as an activity of the total process, and it may be executed in another workflow engine. Figure 4(b) shows a Kepler process that will invoke the workflow service. There are four parameters in the kepler process, "Constant1" represents the mode of service invocation while "0" is asynchronous and "1" is synchronous;

"Constant2" is the goal's name; "Constant3" is goal parameters' name which separated by commas; "Constant4" is goal parameters' value, such as the string "2,3" represents there have two departments and each department has three person. The "WSWithComplexTypes" component of this Kepler process has been bound to the encapsulated engine service's WSDL link as the dashed line① shown in Figure 4(b). When the Kepler process needs to call this product acceptance process, it will start the workflow service and then execute the process, and at the same time it will deliver the parameters that the process required into the workflow engine. As the dashed line② shown in Figure 4(b), after the completion of the product acceptance process, this workflow execution server will handle and return the process' result according to the requirements for such as mode of invocation and so on, and then transfer the control back to Kepler engine and continue to execute the next activity of the total process.

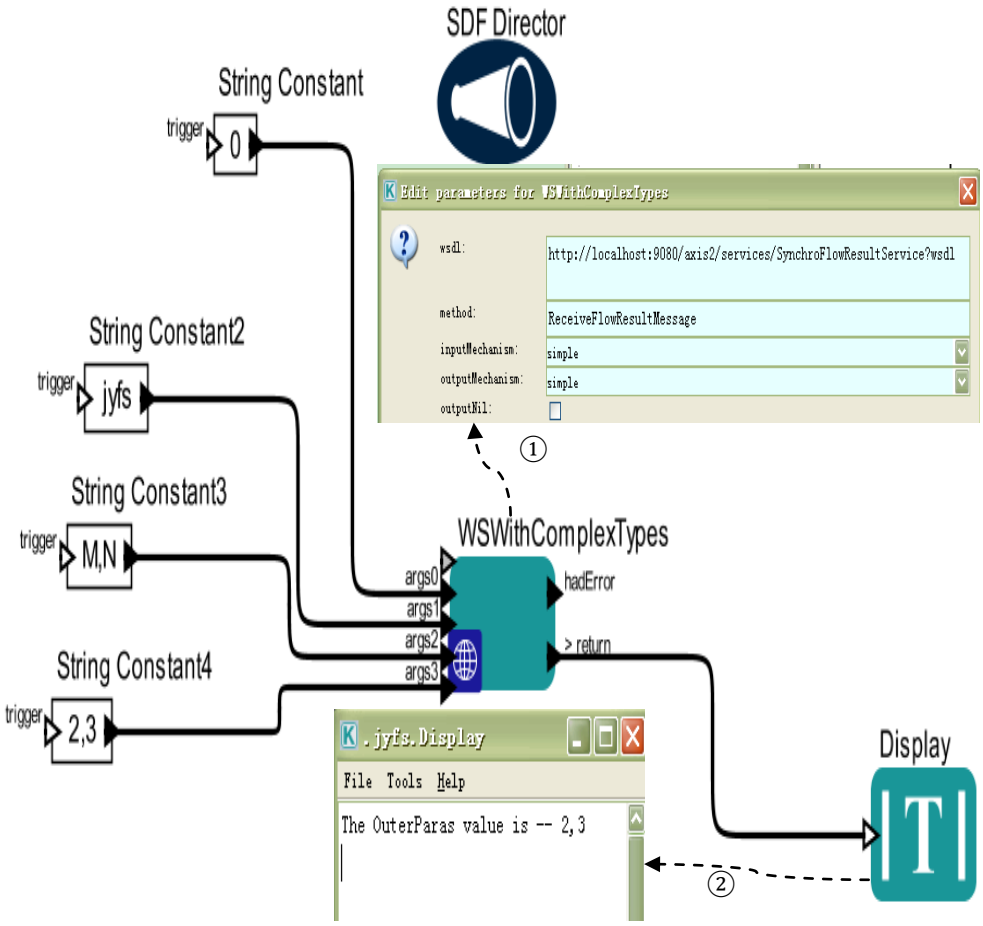


Figure 4(b). Kepler Invoke the Product Acceptance Process

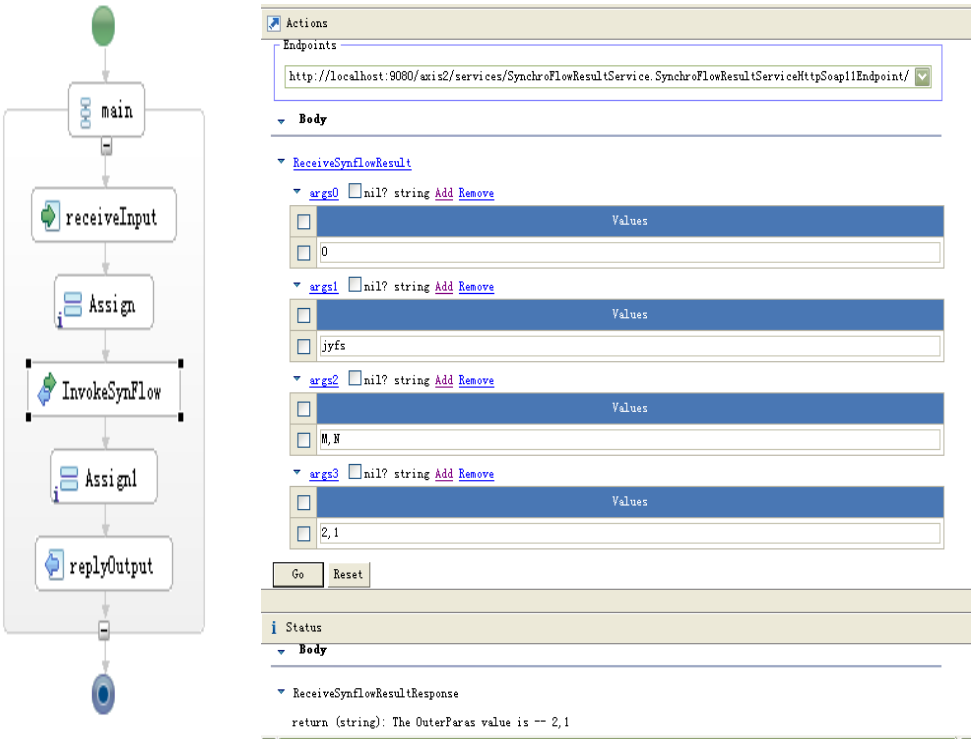


Figure 4(c). ODE Invoke Product Acceptance Process

Figure 4(c) shows a BPEL process that will invoke the workflow service. In this BPEL process, there has an activity called "InvokeSynFlow" which has been bound to the link of the encapsulated workflow service. The BPEL process assigns the process' arguments to the service's input parameters through the "Assign" activity to start a certain process instance which belongs to the workflow service, and the workflow service will return the result through "Assign1" activity after the completion of the Product Acceptance Process.

It can be seen from the above examples that a workflow service could be invoked by two different workflow systems, the design of workflow interfaces and encapsulated method of the workflow service are reasonable and feasible.

5. Conclusions and Future Work

In this paper we present a general solution of workflow interoperability and sharing at the level of heterogeneous workflow processes' integration based on Web Service. In order to ensure the consistency of business information, improve the workflow system's scalability and enhance the compatibility with the combination of heterogeneous processes, we focused on the extension of workflow system's interfaces and the method of workflow service's

encapsulation while the workflow interoperability's standards were explained according to WFMC.

The variety of control, data elements, resource and other factors will continue being developed for various domains and guided by those domain's needs. Therefore, the process services' planning and selection, as well as the uncertainty and reliability of input/output parameters in a distributed environment are a research goal in the near future.

Acknowledgements

The presented work is partially supported by the National Science and Technology Support Program (2012BAK17B09), National Science and Technology Major Project of China (2012ZX01039-003 and 2012ZX01039-004-01-3).

References

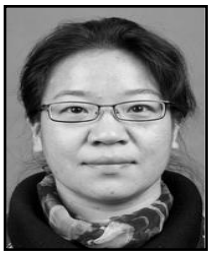
[1] V. Curcin and M. Ghanem, "Scientific workflow systems-can one size fit all", Proceedings of the Cairo International Biomedical Engineering Conference, **(2008)** December 18-20, Cairo, Egypt.
[2] D. Hollingsworth, "The workflow reference model", Workflow Management Coalition, **(1995)**.
[3] B. Yang, K. Yan, J. S. Jiang and G. Y. Hu, "Web Services architecture oriented cooperative workflow model", Computer Engineering and Design, vol. 3, no. 32, **(2011)**.
[4] S. D. I. Fernando and A. C. Simpson, "Towards a formal framework for workflow interoperability", Lecture Notes in Computer Science, vol. 5387, **(2009)**.
[5] G. Pavlin, M. Kamermans and M. Scafes, "Dynamic process integration framework", Toward efficient information processing in complex distributed systems, Informatica, vol. 34, **(2010)**.
[6] T. Kukal, T. Kiss, G. Terstyanszky and P. Kacsuk, "A general and scalable solution for heterogeneous workflow invocation and nesting", WORK'08 Proceedings of the 3th on Workflows in Support of Large-Scale Science, **(2008)** November 17, Austin, USA.
[7] A. Alqaoud, I. Taylor and A. Jones, "Publish/Subscribe as a model for scientific workflow interoperability, WORKS'09 Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science", **(2009)** November 16, Portland, USA.
[8] D. Tang, R. Z. Sun, Y. Xiang and G. Yuan, "Interface design of heterogeneous workflow interconnection based on Web service", Journal of Computer Applications, vol. 6, no. 33, **(2013)**.
[9] X. Guo and W. Huang, "Grid-based Region Management System based on SOA architecture research", Information Engineering and Electronic Commerce, 2009, IEEC'09, International Symposium on **(2009)** May 16-17, Ternopil, Ukraine.
[10] P. Jiang, Y. Ding, L. Gao, X. Y. Shao and Y. D. Shen, "A distributed workflow management system for collaborative product development", Proceedings of 2010 IEEE the 17th International Conference on Industrial Engineering and Engineering Management, **(2010)** December 7-10, Macau, China.
[11] P. X. "A Design of Platform System Supporting Dynamic Workflow Interoperation and Enterprise Application Integration, Industrial Control and Electronics Engineering (ICICEE)", 2012 International Conference on. **(2012)** August 23-25, Xian, China.
[12] K. Plankensteiner, R. Prodan, M. Janetschek, T. Fahringer, J. Montagnat, D. Rogers, I. Harvey, I. Taylor, A. Balasko and P. Kacsuk, "Fine-Grain Interoperability of Scientific Workflows in Distributed Computing Infrastructures. Journal of grid computing", vol. 3, no.11 **(2013)**.
[13] F. Zhen, M. Liu and M. Y. Dong, "SOA message-oriented middleware based system integration method for business process", Computer Integrated Manufacturing Systems, vol. 5, no. 15, **(2009)**.

Authors

Gang Yuan, working toward the PhD degree in the College of Information and Electrical Engineering, China Agricultural University. Her current research interests include workflow technologies and applications, business process management and web services.

Rui-zhi Sun, received his PhD degree (2003) in Tsinghua University. He is a Full Professor with the College of Information and Electrical Engineering, China Agricultural University. His research interests include computer network and applications, workflow management and cloud computing.

Yong Xiang, received his PhD degree (1998) in Tsinghua University. He is an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include ad hoc network and computer supported cooperative work.



Yin-xue Shi, working toward the PhD degree in the College of Information and Electrical Engineering, China Agricultural University. Her current research interests include business process management, workflow technologies and applications.