Distributed Workflow Architecture Based on Flexible Data Management

Yinzhou Zhu[1] and Baolin Yin[2]

[1,2] National Laboratory Software Development Environment, Beihang University, Beijing 100191, China
{zhuyinzhou, yinbaolin}@nlsde.buaa.edu.cn

Abstract

In recent years, distributed workflow management technology attracted much attention for the requirement of rapid development of process oriented software systems. However the current workflow management systems do not have sufficient capabilities in controlling data stream between the workflow engine and applications, decreasing the reusability of the workflow components. Therefore we enhance the basic data model of the distributed workflow system, and propose an architecture based on a flexible data mapping model to improve the reusability of the workflow components in distributed workflow management. A real-world scenario which is built and implemented based on this architecture is shown to prove the effectiveness and usefulness of the architecture.

Keywords: *Workflow Management, Business process, Distributed systems, Data mapping, Reusability*

1. Introduction

With the expansion of global business, more and more enterprises expect to construct and improve their business processes rapidly and frequently [1]. Under such circumstances,, distributed workflow management systems (DWFMSs) attracted much attention because of their comprehensible and flexible business process model. Software development based on DWFMS often used a top-down design method, and followed a stepwise refinement process. This process need close co-operation between business process analysts and programmers [2]. In today's businesses, however, when the workflow requirement was more and more driven by the customs' demands, the workflows have to be developed faster and more agile than before. This implies that the architecture of distributed workflow development should be improved so that the capabilities in controlling data stream between workflow engine and application could be enhanced to simplify the configuration of the workflow applications.

In a typical DWFMS, the first step of workflow development is to define the process model with the help of the business process analysts who master the basic computer knowledge. They have to define the workflow activities, the resources on which the activities are worked, the applications that will be executed by the resources, and the start-conditions that describe when an application could be executed. Then the programmers will complete the workflows by converting the descriptions to the programs and coding for low level functions [2]. In many cases, the applications of the business processes are similar to each other; only the input/output data objects and the user permissions of the application are different. This opens the possibility of developing the workflow by applications that was extended from standard components, which will save programmers much time and labor. A important thing to note in workflow developing is the interface of the components, as it should be simple and flexible enough to learned by business process analysts

To support the development of the application that have a common user interface, the component with application-level granularity have more advantages than the web-service which were adopted in most traditional WFMSs [3]. To keep the simplicity and the automation of the process of the workflow development, the interface of the components must be unified and explicitly defined. To mapping complex data structure between the workflow engine and components, the data model of the workflow must be flexible and easy to operate. Thus we choose the standard relational table as the basic data type in our DWFMS, for its simplicity and powerful expressiveness. To support this data model in the DWFMS, it have to improve the infrastructure of distributed data storage and transmission. To support the flexible integration between the workflow engine and the components, the DWFMS should extend the data mapping mechanism based on flexible data mapping model. A graphic design tool for data mapping is also required to help users develop the workflow more easily and rapidly.

In the EasyWork project, the support of the relational table and the flexible data mapping mechanism is implemented for distributed workflow development. Along the lines presented in this paper, the data mapping model and the component interface specification have been designed to model and specify flexible data mapping. Based on the DWFMS in the EasyWork project, related infrastructure have been implemented to select the components, configure the applications, integrate the applications with workflow engine, and test the activities of the workflow.

This paper focuses on the data mapping model and the component interface specification in the EasyWork project. A real-world workflow application developed by the project is discussed to illustrate the use of the flexible data mapping technique.

2. Related Works

In the past decade, workflow management technology has played an important role in the fields of business process management and scientific computation [4, 5]. The emphasis has mainly been on distributed workflow management in scientific computation environments [5]. With the advent of rapid evolution of the business process in large enterprises, however, distributed workflow management is attracting much attention in business process management field these days.

Muthusamy *et al.*, developed a flexible and distributed platform to develop, execute, and monitor business process [6, 7]. This platform is based on web services, and utilize a distributed content-based publish/subscribe overlay which could work effectively in heterogeneous environments. The system focus on the service discovery, composition, and the efficiency of the remote service call, but the discussion of data flow management between the workflow engine and services is not sufficient for rapid workflow development.

Khalaf and Leymann present a BPEL fragmentation covering data and explicit control dependencies, and an approach to handle fragmenting loops and scopes [8]. The system utilizes the BPEL workflow engine and web services, which are difficult to construct the applications with a user interface.

Hamann *et al.*, present a migration data meta-model for business processes with the ability for runtime migration, which enhance the flexibility of the distribution of the ad-hoc workflow [9]. The prototype is applied to WS-BPEL and XPDL processes, which is not sufficient for rapid workflow development also.

Besides, most of common business process management systems, such as Staffware, WebSphere, FLOWer, COSA, or workflow model language, such as XPDL and BPEL4S, only support basic data types when communicating with the applications [10].

Our work distinguishes itself from these other approaches by enhancement of the basic data model of the workflow engine and the components, which could help to assemble the workflows in more flexible ways.

3. Architecture

The architecture of the EasyWork system, which is shown in Figure 1, aims to define, execute and monitor the workflows for the cross-regional enterprises which often have several highly autonomous subsidiaries. The network architecture of the EasyWork system is a hybrid structure, which is based on the peer to peer network and the client/server framework. There are two kinds of nodes in the EasyWrok network: EasyWork Server node and the EasyWork Client node. The EasyWork Servers are the basic nodes which are used to store and dispatch the distributed workflow instances. The relationships between EasyWork Servers are symmetrical, while the EasyWork Clients are client nodes of the EasyWork Server. The EasyWork Platform is installed on every EasyWork Server, and offer the access interface that allows users to get and do their job in remote EasyWork Clients though a standard web browser.

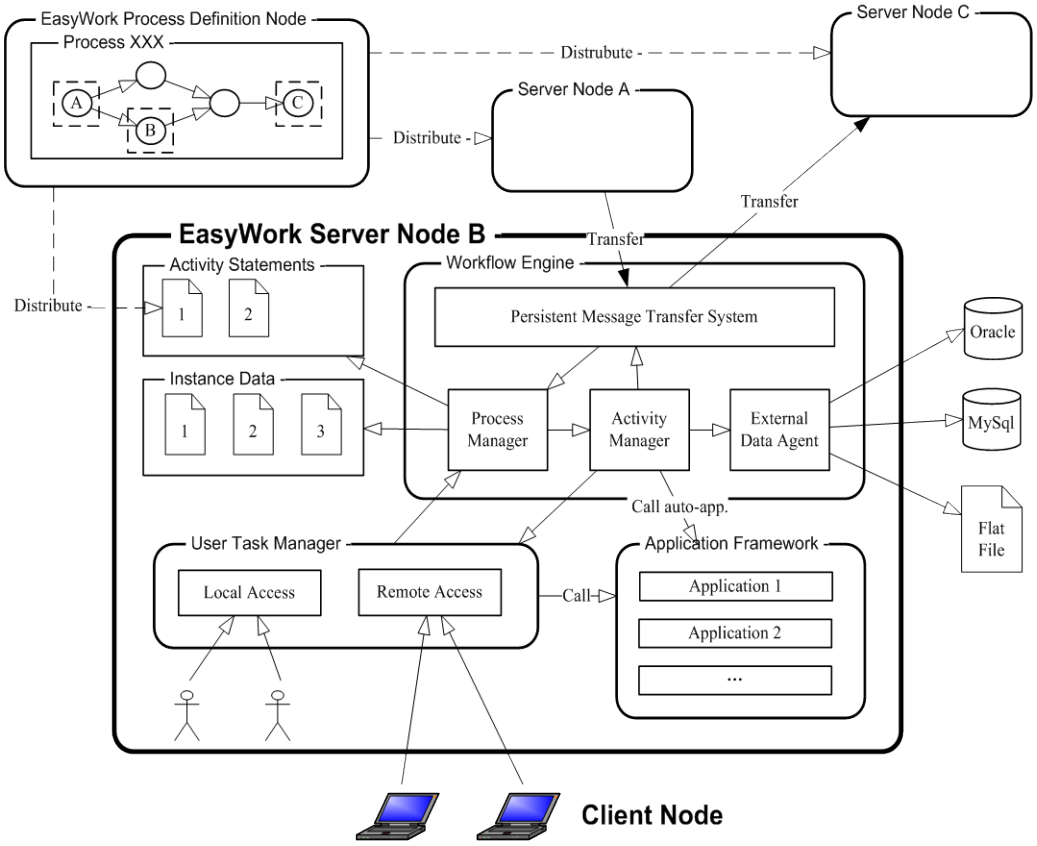


Figure 1. The Architecture Diagram of the EasyWork system

EasyWork Process Definition Server is a kind of EasyWork Server on which the workflow definition tool is installed. The workflow definition tool is used by workflow administrators to define, compile and deploy the distributed workflow. After a workflow model has been designed in the Process Definition Server, it will be split into several segments by activities,

and be compiled to configuration files. These configuration files will then be distributed to EasyWork Servers which are defined as the computing resources of the activities.

The EasyWork Platform on EasyWork Server is composed of three main parts: the workflow engine, the user task manager and the application framework. The workflow engine is used to receive, store, and dispatch the workflow instances, and invoke the application to process the tasks of the activities. The user task manager offers the access interfaces of the workflow system to users, shows the task-lists, and communicates with the workflow engine to process user commands. The application framework is a set of applications which are created based on the components with application-level granularity. These applications, which are invoked by the workflow engine to process the tasks of the activities, are executable programs, such as the executable binary files of the operational systems of Microsoft Windows or UNIX, or the web pages that could be interpreted by http server or client browser.

4. Process Model

EasyWork project contains two main parts: the workflow design tools and the distributed workflow engines. Business process analyzer utilizes the workflow design tools to define, test, deploy, and monitor the workflow. And the users specified in the workflow receive and handle their jobs though workflow engines which were distributed over all workflow network. The process model defined in workflow design tool contains two levels: the upper level is the business level based on the directed graph of the activities; and the lower level is the execution level based on the sequences joined by applications, as shown in Figure 2. The reason of splitting the process model into two levels is the different nature of the process model in two levels. The process model in the business level is easy to understand and analyze from the business perspective, and the process model in the execution level specifies the details of the applications.

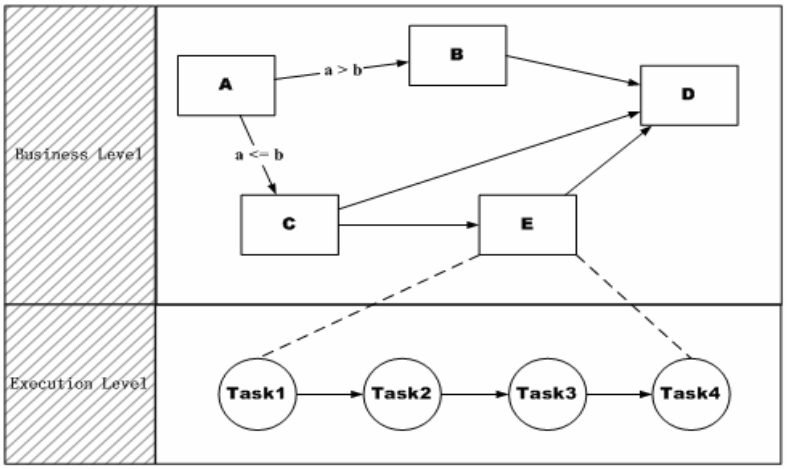


Figure 2. Process Model Diagram in EasyWork Project

4.1. Business Level

The process model in the business level is made up of six elements: activities, transitions, resources, users, start conditions, and branch conditions. Every node in the directed graph represents an activity which will be executed on a computer resource which is owned by one or more users. The lines between nodes represent the transition path between activities. Each

transition path may bind to a branch condition which is a logic expression composed of the common operators, constants and the workflow data. Each activity has a start condition, which is composed of the logic operators and the identifiers of the preceded activities, to specify when to start the applications of the current activity. When the user complete the sequence of the applications of an activity, the workflow engine on this computer will test all the following branch conditions to find out which branch conditions are met. If a branch condition is met, the workflow instance will be transferred to the computer where the branch condition links to. After a workflow instance is transferred to a computer node, the node will test the start condition of the activity to find out whether the required workflow instances are all achieved, and if yes, the activity will be marked as an undo-job in the job list of the specified users.

With the help of the process model in business level, the workflow design tool could support all typical workflow model patterns, such as And-Split, And-Join, Or-Split, Or-Join. Besides, this model decreases the number of the operator types and simplify the relation between the activity and the computer resource.

4.2. Execution Level

The jobs specified in the activity are composed of the applications which are executable programs, such as the executable binary files on the operational system of Microsoft Windows or UNIX, or the web pages that could be interpreted by http server or client browser. These applications is produced from configuring the components that chosen from the component library. The library is composed of a set of basic standard components, such as form, grid, report construction, or email delivery; and many domain specific components, such as the components used in office domain, submit/approval domain, financial domain. All components in the library have the unified interface to configure and execute. When defining the process model in execution level, it first has to configure the input and output data and the execute conditions of the components to create executable applications, and then add the configured applications to task sequence of the activity. When the activity is started, the workflow engine will execute the applications in the task sequence one by one, mapping the suitable workflow data to the input of the application, and mapping the output of the application back to the workflow data.

Based on process model in execution level, the DWFMS could handle the workflow data more flexible, reducing the coupling between the workflow engine and the applications, making the components and the workflow data more independently, increasing the reusability of the components. Combined with the flexible data mapping mechanism which simplify the data mapping operation, the EasyWork project can help the business process analysts break out of their dependence on the programmers.

5. Flexible Data Mapping

Flexible data mapping mechanism, which is the key technique for binding the applications to the workflow, is used to define the data mapping relationships between the workflow data and component data, and to convert the data schema based on these relationships when the workflow engine invoke the applications.

5.1. Workflow Data Bus

Workflow Data Bus is an abstract layer used to store and offer the workflow data in EasyWork project. It manages all the business data transmitted in the workflows. When a workflow instance has been started, the Workflow Data Bus transfers the instance data from

one activity node to another activity node in the workflow network. When running an application, Workflow Data Bus offers the data which required by the application, and store the return data from the application when it is finished, as can be seen inn Figure 3.

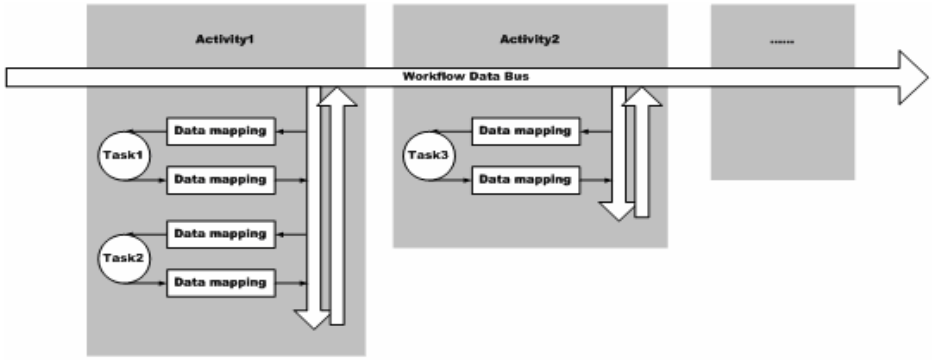


Figure 3. Data Flow in the Workflow Instance Execution

The data model of Workflow Data Bus is based on relational table. All workflow data is represented in the form of relational table. Also, in the definition of the component interface, the data model of the input data and output data of the components are based on relational table. This approach opens the possibility of converting the data schema of the workflow data when mapping workflow data to component data so that the new data schema of the workflow data could meet the component data requirement. The same goes for the mapping from component data to workflow data after the application have been finished.

5.2. Data Mapping Specification

The primary goal of the flexible data mapping mechanism is to support the conversion of the data schema between the workflow data and the component data. In order to make it easier to use and learn for the business process analysts, a SQL-like language has been used to specify the flexible data mapping. A mapping description is like this:

*dest_data [+]= **SELECT** src_field_list **FROM** src_data_list [**WHERE** condition]*

In this line, *dest_data* represents the variable to which the converted data is assigned in the mapping operation. The list *src_data_list* is composed of one or more source variables which are from the workflow data or component data. The list *src_field_list* is composed of one or more constant or column names, which are in the tables of the source variables, to specify the columns to be retrieved. The *condition* is a logic expression to select the proper tuples from the sources variables. When performing a mapping operation, the tables in source variables are joined together to take the Cartesian product, and then select the tuples though the logic expression *condition* and retrieved the columns that specified by *src_field_list*. The result data will be assigned to *dest_data* in two ways: rewrite or append, which are represented with the operators "=" and "+=" respectively. The rewrite mode will clean the old data of *dest_data* and assign a new table to it. And the append mode means the old data will be retained and new data will be appended to it.

3. Component Interface Specification

Each component on which the applications are based must give a specification about its basic information and data schemas, which is called component interface specification, to help the business process analysts understand how to configure the component and how to specify the data mapping between workflow and component. The workflow design tool in the EasyWork project reads the component interface specification files and show them in a visual way. Below a demonstrative component interface specification is given.

Table 1. Demonstrative Component Interface Specification

Specification	Annotation
Component_Name: {	# Basic information block
title: **STRING**,	# Component title
desc: **STRING**,	# Component introduction
class: **STRING**,	# Component classification
input: [*var1, var2, …*],	# Input variables of the component
output: [*var1, var2, …*]	# Output variables of the component
},	
var1: {	# Data information block
title: **STRING**,	# Variable title
table*:* {	# Data schema block
field1: [*fieldType, setting*],	# Field setting
field2: [*fieldType, setting*],	
…	
},	
canModified: **BOOL**,	# Whether the data schema can be modified
desc: **STRING**	# Variable introduction
},	
…	# Other data information blocks

A component specification file is composed of many information blocks. The first block is basic information block, and others are data information blocks. The basic information block consist of the component's name, title, classification, a simple introduction, and input and output variables. The data information blocks specify the variables of the component one for one. And each data information block consist of the name, title, initial data schema, and a simple introduction of a variable.

With the help of the component specification files, business process analysts could configure the applications and its data mapping rules in the workflow design tool. And when the applications are invoked, the workflow engine will automatically mapping the data between workflow and applications.

6. Case Study

To illustrate the effectiveness of the business process development with the architecture presented the EasyWork project uses on a real-world scenario as input: the Freshwater Quality Monitoring scenario (FQM).

6.1. Background

Every year, the National Oceanic Administration of China needs to investigate the freshwater quality in the coastal areas, collecting the information, and provide a summary report. There are over 100 observation stations located in different parts of the country, which help the administration investigate the water quality. The FQM process is started by the officer at the headquarters, and the lists of the test items are sent to the observation stations. Then the observation stations investigate the water quality according to the lists they received, fill out the investigation forms, and send them back. The head office collects these responses, calculates the related indexes, and provides the summary report at last.

6.2. Data Mapping Specification

The primary goal of the flexible data mapping mechanism is to support the conversion of the data schema between the workflow data and the component data. In order to make it easier to use and learn for the business process analysts, a SQL-like language has been used to specify the flexible data mapping. A mapping description is like this:

The FQM process was implemented based on the DWFMS in EasyWork project. The implementation is divided into four steps: (1) Define the activities and the organizations. (2) Choose the proper components and configure them to create the applications for each activity. (3) Define workflow variables and consider the data mapping specifics. (4) Deploy the distributed workflow in the DWFMS network.

Table 2 shows some important attributes of the activities on FQM process. There are 134 activities, two of them are executed by head office to start and summarize the investigation, and others are distributed to 132 observation stations located along the east coast to fill out the investigation forms.

Table 2. Activities on the FQM Process

ID	Name	Purpose	User	Applications
A1	Start	Start the Investigation	Head office	Report Setting
B1 ~ B132	Submission	Fill out the investigation form	Observation Stations	Investigation Form
C1	Summary	Calculate the index and show the report table	Head office	Index Calculation
				Report Grid

The application used in the process is described in Table 2 and Table 3. There are four applications extended from three basic components. The components *Form* and *Grid* are used to construct GUI applications which can present the data to users and get the input from users. The component *Calculator* is used to construct the computational applications which are executed without any user input.

Table 3. Basic Components and Extended Applications used in the FQM Process

Basic Components	Introduction	Application
Form	Show a data input form based on field names, field types, and constraints	Report Setting
		Investigation Form
Grid	Show a data grid table based on the header setting and the data	Report Grid
Calculator	Calculate the result list based on input variables and formulas	Index Calculator

Table 4 shows the main workflow data in this process. The variables *Form* and *Collect* have the same data schema to store the information of the 376 test items of the investigation form. The difference is that the variable *Form* just has only one tuple while the variable *Collect* has multiple tuples. Since the DWFMS in EasyWork project could offer a buffer for each activity to store the temporary workflow data, it is not necessary to set up a central database system for this workflow, so we just use a local database in node C1 to backup the summary reports.

Table 4. Main Workflow Variables on the FQM Process

Variables	Introduction	Data schema
Form	Store the test names and values	(*p1, p2, p3, …, p376*)
Collect	Collect all the test values in FQM process	(*p1, p2, p3, …, p376*)
Result	Store the index name and values that shown in the summary report	(*index_name, index_value*)

The workflow model and data mapping specifics can be seen in Figure 4. First of all, three main workflow variables are initiated in activity A1. And then these variables are transferred to every observation station though the Workflow Data Bus. From activity B1 to B132, each of them save the information of the submitted investigation form to the branched variable *form*. All these branched variables are assembled in activity C1, and are presented as the variable *act.form*, where *act* is the name of the preceded activity from where the variable is sent.

From the above process which has been worked in the real world, we can see many advantages of the architecture presented in this paper. Compared to most traditional DWFMSs which often only manage the simple status data, our DWFMS chooses a more flexible data management model to manage complex data. As a result, we can save much time and labor that is used to code for controlling the data flow to connect the applications to the workflow. For example in above process, we didn't write any code for the applications. On one hand, the reusability of the components are improved, and on the other hand, this approach enhances the control of the data for process analysts, which could reduce the probability of errors in the workflow design.

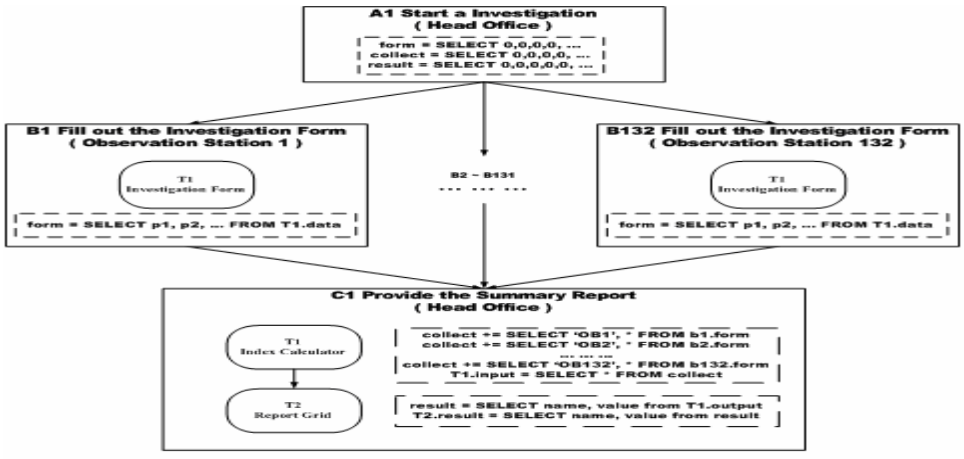


Figure 4. Distributed Workflow Model and Data Mapping Specifics for the FQM Process

Compared to the object-oriented data models, relational table is simpler and more universal, and easier to operate when sharing data with databases. The operation of the relational table is based on relational algebra which is strong in logic and systematism, so the relational table is more appropriate to the occasion of converting the data model.

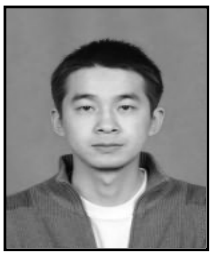
7. Conclusions

By the use of flexible data mapping mechanism, increasing the reusability of the workflow applications is possible. The data model of the architecture is based on relational table, and a SQL-like language is chosen to specify the flexible data mapping. Compared to traditional DWFMS, this approach enhances the power of the data flow management, so that save much time and labor on developing similar applications. The data mapping language is easy to learn, which make business process analysts less dependent on application programmers, and improve the efficiency of the workflow development. Besides, as the data flow is more controlled by the process analyzer, the applications could be designed in a more universal way. In future, the database operation will be integrated into the flexible data mapping mechanism to provide a more powerful method of the data flow controlling.

References

[1] J. Sinur and J. B. Hill, "Magic Quadrant for business process management suites", Gartner RAS Core research note, **(2010)**, pp. 1-24.
[2] C. Ouyang, M. Dumas, W. M. P. Aalst, A. H. M. T. Hofstede and J. Mendling, "From business process models to process-oriented software systems", ACM transactions on software engineering and methodology (TOSEM), vol. 19, no. 1, **(2009)**, pp. 1-37.
[3] G.-J. Jin, B.-L. Yin and Q.-Y. Zhao, "Reusability enhancing method for integratable software components with application-level granularity", Computer Integrated Manufacturing Systems, vol. 17, no. 2, **(2011)**, pp. 425-432.
[4] W. M. van der Aalst, "Business Process Management: A Comprehensive Survey", ISRN Software Engineering, **(2013)**.
[5] E. Deelman, D. Gannon, M. Shields and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities", Future Generation Computer Systems, vol 25, no. 5, **(2009)**, pp. 528-540.
[6] V. Muthusamy, "Flexible Distributed Business Process Management", University of Toronto, **(2011)**.
[7] S. Ganesan, Y. Yoon and H.-A. Jacobsen, "NIÑOS take five: the management infrastructure for distributed event-driven workflows", Proceedings of the 5th ACM international conference on Distributed event-based system, DEBS, **(2011)**.
[8] R. Khalaf and F. Leymann, "Coordination for fragmented loops and scopes in a distributed business process", Information Systems, vol. 37, no. 6, **(2012)**, pp. 593-610.
[9] K. Hamann, S. Zaplata and W. Lamersdorf, "Process instance migration: Flexible execution of distributed business processes", Software Services and Systems Research-Results and Challenges (S-Cube), 2012 Workshop on European, IEEE, **(2012)**.
[10] N. Russell, A. Ter Hofstede, D. Edmond and W. van der Aalst, "Workflow data patterns: Identification, representation and tool support", Conceptual Modeling–ER 2005, **(2005)**, pp. 353-368.

Authors



Yinzhou Zhu received his M.S. in 2006 and currently is a Ph.D. candidate, both in Computer Science and Engineering Department at Beihang University (Beijing). His research interests include workflow management, enterprise application integration and distributed systems.

Baolin Yin is a Professor in Computer Science and Engineering Department at Beihang University (Beijing), and is currently head of the State Key Laboratory of Advanced Software Development. He received his Ph.D. in Artificial Intelligence in the University of Edinburgh (1984).His research activity is concerned with distributed systems, workflow management, image identification, and computer security.