Design and Implementation of a Flexible Workflow Management System

Yubin Guo
College of Informatics, South China Agricultural University, Guangzhou, 510642 China
Email: guoyubin@scau.edu.cn

Zeye Cai [1], Zewei Lin [1], Ximing Li [1, 2*]
[1]College of Informatics, South China Agricultural University, Guangzhou, 510642 China
[2]Department of Computer Science, Ben Gurion University of the Negev, Beer-Sheva, 84105 Israel
zycai, zwlin@scau.edu.cn, ximing@post.bgu.ac.il

Abstract—**In modern society, flexible workflow is necessary for enterprises which will enable them to keep up with market variations and new technologies quickly, and to improve the whole efficiency of the enterprise. Firstly, this paper presents a formal application model of flexible process for Flexible Workflow Management System (or FWMS in short). Then, we describe the prototype in detail and give the architecture and functional modules of it. Moreover, the prototype is implemented practically with Struts, Hibernate software framework as a web application. We also give a flexible homework assignment system as a application of the prototype system which can support personalized homework assignments and communications.**

Index Terms—**Flexible workflow system, work program, Business Process**

I. Introduction

Flexible workflow management technology is one of the core technologies to fasten information system development in distributed and dynamical environments. It enables enterprises to deal with changes of workflow definitions resulting from variations of market, new technologies and new laws quickly, and to improve entire efficiency. Recently the need for enhancing flexibility and the integration of applications in heterogeneous environments has been a focus in both academy and industry.

Some representatives in recent researches of the flexible workflow technology include the ADEPT2 Project [1], the Case Handling Method [2] and DECLARE [3]. Among them, ADEPT2 focused on the process structure of dynamic adjustment; the Case Handling Method was strongly based on data as the typical product of these processes. And DECLARE is a proto-type of a WFMS that uses a constraint-based process modeling language for the development of declarative models describing loosely-structured processes. And ADEPT2 and DECLARE [4, 5, 8] have been gradually put into application. Besides, the technique of Agent [6] and the rule of ECA [7] have been introduced to improve the flexibility of workflow system also. Gang Ye et al. [8] used workflow technology into testing for improving the automation of spacecraft software testing. And a core scheduling algorithm was implemented and analyzed in their dedicated workflow engine. Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid is discussed in [9]. M. Reichert and J. J. Li have given some comparisons of many kinds of research methods respectively [10], [11], both of which have drawn a wonderful conclusion of the Flexible Workflow System's theoretical research and implementation methods.

Comparatively, nowadays studies on theory of Flexible Workflow are much more than the researches on implementation. Works are rare on how to design and implement a flexible workflow management system using modern design patterns and software architectures. In this paper, definition of a relatively universal flexible workflow model is given at first. And then a corresponding model, Flexible Work Process Model (FWPM, in short) is defined for implementation. Based on FWPM, system design and implementation of a prototype system is presented. And an application example, the Homework Management System, is given to illustrate that the design is in practice. This is a personal homework management system, homework can be assigned personally, and communications can be added by related teachers and students when certain homework is being processed.

The paper is organized as follows: Section 2 introduces preliminaries and the flexible workflow model. Section 3 proposes architecture of the flexible workflow management system, FWMS in short, and key techniques of implementation; Section 4 illustrates usability of the design by an application, a homework management

Manuscript received April 1, 2013; revised June 4, 2013; accepted July 1, 2013.
This work was partially supported by GuangDong Province Undergraduates innovating projects (1056411054). National Science Foundation (61103232, 61272402, 61202294), Guangdong Science Technology Plan Project (2011B090400325) and Guangdong Province Nature Science Foundation(10351806001000000, 10151064201000028).
Contact author: ximing@post.bgu.ac.il (Li Ximing)

system. And Section 5 concludes and puts forward future works.

II. Preliminaries and Flexible Work Process Model

Flexibility of workflow means the dynamic generation and modification on definition of process instances during execution. Meanwhile, process instances can be abstracted into templates for the coming processes with similar pattern. In this section, a universal concept of workflow model, process, is introduced at first, and the Flexible Work Process Model is defined for implementation accordingly. For more details on the definition of process, readers are referred to reference [12].

Definition 1 (Process) a process is a 3-tuple $P= (A, <, \lhd)$ with:

(1) A is a set of activities and processes.

(2) $<$ serves as the sequence relation among activities and processes. To $\forall a_1, a_2 \in A, a_1 < a_2$ means a_1 must be executed before a_2. It is a partial relation describing executing order of elements in A.

(3) $\lhd$ is a selection on A. There exists a selective predicate $\lhd(a_1, a_2)$, and if the predicate is true, activity a_1 will be executed, else activity a_2 will be carried out.

In Definition 1, only two relation, sequence and selection are defined in a process. In general, there are three types of logic relation among parts of a process including sequence, concurrency and selection. Two activities are in concurrency if there is no sequence or selection relation between them. Two activities can be executed in any order if they are in concurrency relation.

In Business Process Model and workflow system, relations between activities can also be defined as sequence, AND-split, OR-split, AND-Join and OR-Join etc. It is proved that the two definitions are equivalent [13].

For convenience to implementation, activity is defined as node in FWMS, selection and sequence relation among activities are denoted by arcs from source nodes to destination nodes. Each process is composed of nodes and arcs together.

Similar to general workflow system, in FWMS, activities are basic building blocks in order to construct processes. Some nodes are weaved together to form a process according to business logic. Processes can also be used as blocks, and can be used together with activities and other processes to construct more complex processes.

In FWMs, both activity and process are in two forms, templates and instances. Template is static, which is the definition of activity or process, while instance is a running and dynamic binding of the template. In FWMS, Node Template, Node Instance, Process Template and Process Instance are defined respectively.

Definition 2 (Node Template*)* Node Template is a 5-tuple *NT=<id, R, ER, P, A, L>* with

Id: The overall unique identifier of the node.

R: The set of all possible results of the node, *R={Result, Result …}*, with *Result ={id, Name, Desc}*. In *Result*, *Id* is the overall unique identifier of result while *Name* serves as the result name and *Desc* is description of the result.

ER: The set of roles and users who have the privilege to modify or execute that node.

P: An entrance to business processing of the activity in application. That can be a URL of processing page, or a service call on cloud with proper parameters.

A: Attachment address, the position of the attachment of the node in the system.

L: Remarks.

Definition 2 presents a formal description of activity (and process), and it is an equivalent to elements in A in Definition 1. It shows basic, static information of activity (and process) in detail. In running stage, it must be instantiated as instance. In FWMS, node instances inherent all attributes from their node templates can be executed and modified when we execute the system.

Definition 3 (Node Instance) *Node Instance* is a 3-tuple *NI ={id, NT, S, SR }*

Id: The overall unique identifier of node.

NT: Template of this node. In *NT=<id, R, ER, P, A, L>*, *R*, *ER*, and *A* can be binding to a certain value belong to set in *NT*.

S: The state of the task node with $S \in$ *{working, completed, hung, terminated}*.

NI is an instance of Node Template that the system assigns to the users. It inherits information of Node Template and includes some instantiation information, like specific executor, executing time, specific attachment information and the result of execution. *State* is a dynamic concept, in this system, only four states *working, completed, hung, terminated* are considered. One *NI* must be at one state at a specific time.

Based on the definitions of node template, relation among nodes, we give the definition of *Arc* as follows.

Definition 4 (Arc) *Arc* is a 4-tuple *Arc= {id, F, N, C}* with

Id: The overall unique identifier of arc.

F: The *id* of its source node.

N: The *id* of its destination node.

C: A logical expression and is the control rule for the direction of process flow.

Definition 4 is the implementation version of relations among activities in process in Definition 1. As to sequence relation, $\forall a_1, a_2 \in A, a_1 < a_2$, there is an arc from a_1 to a_2 with the condition being set to "true", that is $\{id_x, a_1, a_2, true\}$. The destination node a_2 can be executed after the source node a_1 of the arc is finished.

For selective relation, "condition" C should be set as selective predicate. For example, let a be the pre-node of selective predicate $\lhd(a, a_1)$, there should be an arc $\{id_y, a, a_1, \lhd(a, a_1)\}$ and another arc $\{id_z, a, a_2, \neg\lhd(a, a_2)\}$. Node a_1 can be executed if a is finished and the selective predicate $\lhd(a, a_1)$ is "*true*". Similarly, node a_2 can be executed if a is finished and $\neg\lhd(a, a_2)$ is "*true*".

It is noticeable that with the definition of arc, relations, like multi-choice, concurrent are very simple to be expressed. Let $\{id_u, a, a_1, \lhd(a, a_1)\}$, $\{id_v, a, a_2, \lhd(a, a_2)\}$, $\{id_w, a, a_3, \lhd(a, a_3)\}$ be 3 arcs, node a_1, a_2, a_3 are multi-choice if more than one condition among $\Delta(a, a_1)$, $\Delta(a, a_2)$ and $\Delta(a, a_3)$ can be set to true at a certain time. Node a_1, a_2, a_3 are concurrent if conditions $\Delta(a, a_1)$, $\Delta(a, a_2)$ and $\Delta(a, a_3)$ are set to "*true*".

FWPM uses arcs to record the relationship between nodes so as to guarantee independence of nodes, and contributes to convenience of modification of the work process structure. As to node template and node instance, relationship between two node templates are similar to the node instances instantiated from them. That is the reason why template and instance of arc not necessary to be defined separately.

With nodes and arcs, process can be defined as follows.

Definition 5 (Process Template) *Process Template* is a *3-tuple PT={Id, AC, AN, Desc}* with

Id: The overall unique identifier of process template.

AC: The set of arcs that the process contains.

AN: The set of node templates that the process contains.

Desc: Description of the process template.

Process Template is a static structure that contains node templates, relationships among nodes and the description information of process. *Process Instance* is a dynamic concept that is a certain binding of a template. When a process instance is running, some node templates must be instantiated. In FWMS, a *process instance* may contain template nodes which are going to be executed. Instance nodes that have been finished or in execution in a certain state belong to the state set *{working, completed, hung, terminated}*.

Definition 6 (Process Instance) *Process Instance* is a 3-tuple *PI={Id, PT, S} with*

Id: The overall unique identifier of process template.

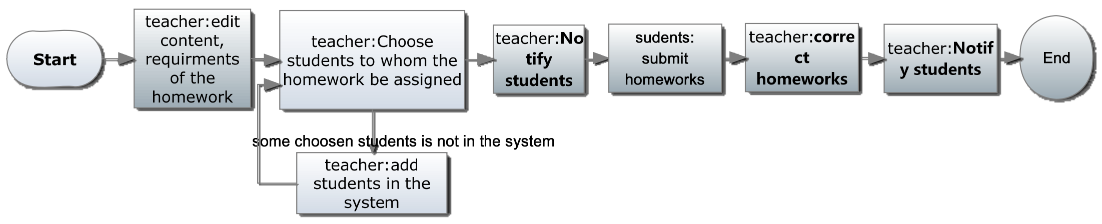
PT: Process Template of *PI*.

S: The executive state of the process, with State $\in$ {*Initializing, working, completed, hung, terminated*}

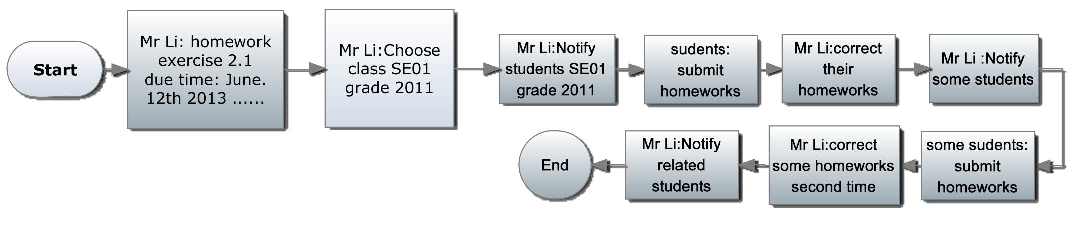
When being initialized, process Instance *PI* is set to its' process template *PT*. And then, proper nodes are chosen to be initialized and executed. The state of *PI* comes from state of its initialized activities. When some activities are *working*, the PI is *working*. When all activities are finished the PI is completed. Similarly, when one of its activities is in *hung* or *terminated* state, the PI is in *hung* or *terminated* state.

Example 1 gives the process of homework assignment. Process template and a certain Process Instance of it are illustrated in Figure 1.

Example1. Figure 1 provides a process of the teacher's homework assignment, in which diagram 1 (a) gives the process template while diagram 1(b) serves as an instance of (a). In general, a teacher must write homework requirements in detail and assign it to a certain group of students and notify them. And students must be added into the course if they are not in. Therefore the process template of homework assignment includes six steps. But as to a certain homework assignment, not all the steps are necessary. In the instance illustrated in Diagram 1(b), "adding students" is neglected because the students of "class SE01 Grade 07" have existed in the system. Then, activities 'Notify students' and 'correct homeworks' are initialized and executed. When correcting homework of students, the teacher finds that some files cannot open properly, so that he (or she) adds new steps temporarily to notify related students, and corrects their homework again. This process is flexible because teacher can add steps temporarily when the process instance is running. Similarly, students can add proper steps when they do their homework, for example, asking for more materials from teacher before submitting their homework, exchanging information with other students. This kind of variation is on process instance without reflecting on template of the process. That is the reason why we define template of process, activity and instance of process, activity separately.


Start
teacher:edit content, requirments of the homework
teacher:Choose students to whom the homework be assigned
teacher:Notify students
sudents: submit homeworks
teacher:correct homeworks
teacher:Notify students
End
some choosen students is not in the system
teacher:add students in the system

(a) Work process Template of homework assignment


Start
Mr Li: homework exercise 2.1 due time: June. 12th 2013 ......
Mr Li:Choose class SE01 grade 2011
Mr Li:Notify students SE01 grade 2011
sudents: submit homeworks
Mr Li:correct their homeworks
Mr Li :Notify some students
some sudents: submit homeworks
Mr Li:correct some homeworks second time
Mr Li:Notify related students
End

(b) an instance of homework assignment

Figure 1. Process of Homework Assignment

III. System Architecture of Flexible Work Process Management System

In this section, system architecture of FWMS is presented, and then the flexible strategy is given.

A. System Architecture of FWMS

FWMS includes four components—work node manager, work process manager, user's task manager and the work process executor, as shown in Figure 2. For each component, function and principle of implementation are presents as follows.

(1) *Work Node manager*. Work node is the basic component of work process, and it is equivalent to Node Template in Definition 2. All work nodes are stored in nodes database with proper classification. Different applications have variant types of work nodes. FWMS offers general node templates and supports users to design new node templates according to their application logic. A certain application where the set of work node are all business activities can be done in the application. Business process means weaving them together according to the business logic.

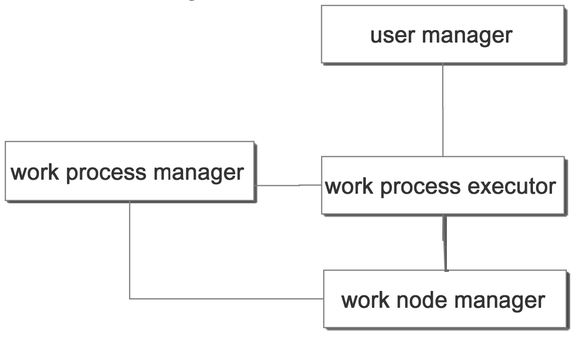


Figure 2. FWPM System Structure

As to software design, each work node is implemented by a URL of processing web page, a service call or other forms of procedure call. All those work nodes must be prepared before they are used in work processes, while work node manager offers management functions of template nodes including creating, editing, deleting and querying of work nodes. Work node manager only manages template nodes. Instance nodes are managed by work process executor.

(2) Work Process Manager Work process manager is in charge of management of work process templates and instances, including functions of creating, editing, deleting and querying of work processes. There is a graphic process editor used to edit work processes. Editing action includes selecting proper node templates from work node database, setting arcs between them to get a work process template and modifying of work processes. Work process manager should interact with process executor, to edit work process instances and illustrate state of work process instances in graphic mode. By work process manager, users can abstract process instances into work process templates when necessary.

(3) User Manager. With this component, user management and authorization are managed. User management includes user verification, login and logout. Authorization includes management of process and authorization in business. Management of process means node, process management. Authorization in business includes roles management in business, assignment of tasks to participants, and the participants can be a certain user or roles (or user groups).

(4) Work Process Executor. Work process executor is the core of FWMS, as illustrated in Figure 3, includes eight main components—execution service, task service, Repository service, management service, history service, identity service, kernel manager and DAOFactory. All components are mounted on a common bus named as ExecutionContext. It is implemented by IOC mechanism in Spring Framework. All services mounted on ExecutionContext, interfaces are defined strictly. When some service is changed (which means implementing class of the service is modified), only related configure files in Spring must be updated accordingly.

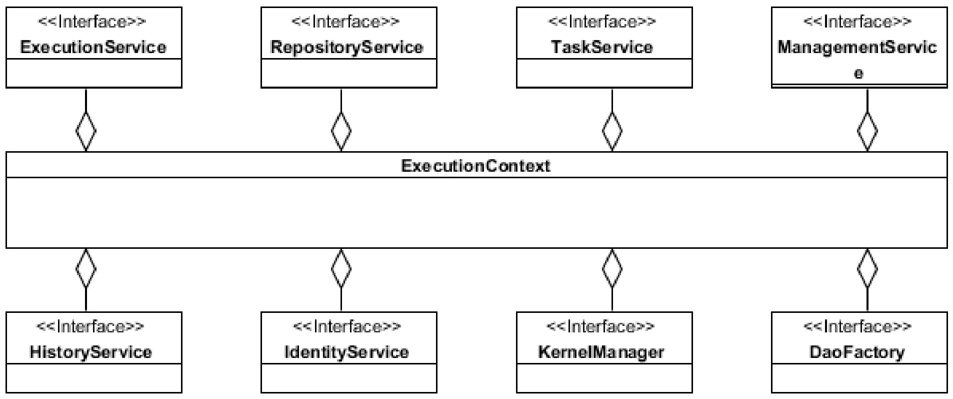


Figure 3. Framework of work process executor in FWMS

In the work process, executor *ExecutionContext* which will be used by the common bus to mount all the services, must be loaded automatically before all services are called. *ExecutionService* provides basic functions of work process execution, including creating, initializing, push process instances on, and set values for process variables. *RepositoryService* is defined for querying, deleting and deploying of process templates, and is an interface to call functions in work node manager and work process manager. *TaskService* is used for creating, querying, submitting, storing and deleting of node instances. It is an interface for business users with proper authorization to change the running process instance by adding or deleting some unexecuted nodes. *ManagementService* is used to provide state querying of process. *HistoryService* is a special interface for querying on finished process instances, node instances and variables and so on. *IdentifyService* is an interface for user management that is implemented by user manager as show in Figure 2. *KernelManager* is a manager to maintain correctness of process instances execution. *DaoFactory* is a tool to stabilize information of process and node instances into database.

As to software architecture, FWMS is a modeled as a MVC system. The view level of FWMS is implemented mostly by JSP+HTML. Users interact with views to login, acquire his or her task list, accomplish his or her task, edit work nodes or processes, create process instances and monitor execution of process instances. EXT JS framework and mxGraph are used for good user experience.

The model of FWMS is process template and process instance. As to management of process templates, controller is deployed on client to improve users operations. Most of management of process instance and work nodes are placed on server side. The controller acts as the intermediary between view and model. It accepts inputs of users and interprets them into an operation on model, and with variation on model, notifies related views to change aspects of their appearance or behavior. Users' Inputs include management of templates of work nodes and processes, executing and monitoring of process instances and so on. Most operations can be translated into operations on model including inserting, deleting, and updating a record in the database.

The system uses XML to exchange information of nodes and processes, and adopts DOM4J as the analysis tool of XML files. The system architecture is implemented on Spring and Hibernate for connecting database. Some other design patterns are used also, like singleton pattern, manager pattern and factory pattern and so on.

B. The Flexible Strategy in FWMS

Flexibility of workflow means definition of workflow can be adjusted or changed during execution in order to adapt to variations of the enterprise. To support flexible workflow, work process executor needs to provide an interface for process transforming temporary. There are two kinds of work process adjusting, modifications on process templates, or on instances. As to process template modification, executing instance can choose to be migrated or not. For instance modification, only the modified instance is effected, and the change must be on effect immediately.

In FWMS, both modification on model and instance are supported. Once a template is changed, a new version of the template is created. When node transforming occurs in a certain work process instance, user can choose to use the new version or old version of its template.

If an instance $PI=\{Id, PT, S\}$ is changed, its work porcess template PT will be copied into the instance, and varation on instance only occurrs on its local work process template, and have no effect on the original template. The instance continue the execution with the new local template.

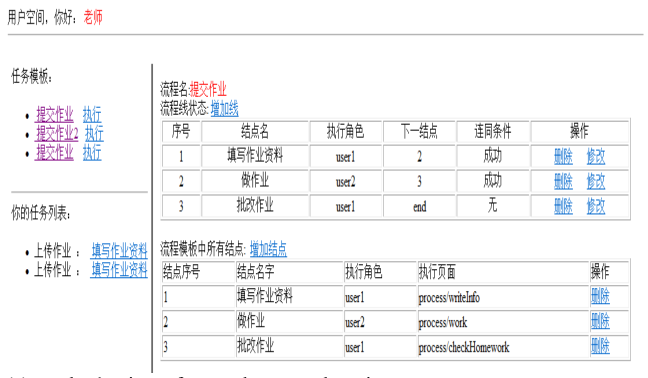
IV. Experiments and Analysis

Traditionally homework management for teachers and students has only one process. Firstly teachers assign homework, students submit their answers. Then, teachers correct students' answers and students browse the teacher's correction. But, sometimes teachers and students need to do some more communications in order to complete the process more efficiently. For example, teacher may want a certain student to submit more materials when he corrects the student's homework. Or a student may want to ask something about certain homework before or after submitting his answer. To accomplish such kinds of communication, methods out of the process are not proper, for coherence of learning.

As an application of FWMS, a flexible homework management system is designed and implemented. There are two roles in the system, teacher and student. Template nodes for teachers are editing homework, assigning homework to students, adding students into the system, correcting students' answer and sending massage to others. Template nodes for students are browsing homework, answering homework, browsing correction and sending massage to others. All the template nodes are business logic from application, and can be weaved together to form a homework process. In FWMS, when homework is assigned, a process instance is Instantiated from a common process template, as shown in Fig. 1 (a). As to a certain homework process, both teacher and student can add additional steps, and all additive steps can be browsed and acquired in detail for they are belong to the same process. Figure 4 illustrates a homework process instance. Figure 4 (a) is teacher's view for arranging a new homework. And Figure 4 (b) is a student view to browse the following steps. The student can add new operations by click proper operation. For this homework, the teacher and the student communicate many times.

V. Conclusion

Flexible workflow is more suitable for the dynamic varying environments of present application system. In this paper, a Flexible Workflow Management System is designed and implemented. A formal implementation model of flexible process is defined at first. Then the architecture and function modules of prototype system are presented. A flexible homework assignment system is given as an application of FWMS.



用户空间，你好：老师

任务模板：

- 提交作业 执行
- 提交作业2 执行
- 提交作业 执行

你的任务列表：

- 上传作业： 填写作业资料
- 上传作业： 填写作业资料

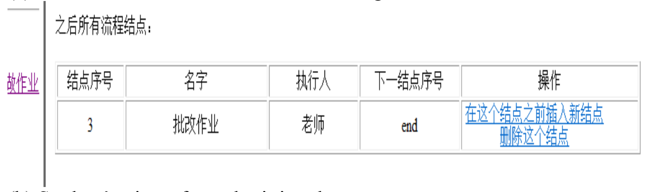
流程名:提交作业
流程线状态:增加线

序号	结点名	执行角色	下一结点	连同条件	操作
1	填写作业资料	user1	2	成功	删除 修改
2	做作业	user2	3	成功	删除 修改
3	批改作业	user1	end	无	删除 修改

流程模板中所有结点: 增加结点

结点序号	结点名字	执行角色	执行页面	操作
1	填写作业资料	user1	process/writeInfo	删除
2	做作业	user2	process/work	删除
3	批改作业	user1	process/checkHomework	删除

(a) teacher’s view of a new homework assignment



做作业

之后所有流程结点：

结点序号	名字	执行人	下一结点序号	操作
3	批改作业	老师	end	在这个结点之前插入新结点 删除这个结点

(b) Student’s view after submitting the answer

Figure 4. Modification of work process in execution

Our future work is to improve the functions and performances of the prototype system. In addition, model checking of dynamic-creating workflow instance is also a challenge to us, and may be solved in the near future.

REFERENCES

[1] Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma et al. From ADEPT to AristaFlow BPM Suite: A Research Vision Has Become Reality [C]. ERBPM’09 Lecture Notes in Business Information Processing, 2010, 43(6): 529-531.

[2] Vander Aalst, M. Weske, D. Grunbauer. Case handling: a new paradigm for business process support [J]. Data & Knowledge Engineering, 2005 53(2):129-162.

[3] M. Pesic, H. Schonenberg, van der Aalst. DECLARE: Full Support for Loosely-Structured Processes [C]. EDOC 2007:287-298 Washington, D. C., USA: IEEE Computer Society, 2007.

[4] W. M. P. van der Aalst, M. Pesic, H. Schonenberg. Declarative workflows: Balancing between flexibility and support. CSRD 2009 23:99-113.

[5] A. Lanz, M. Reichert, P. Dadam. Making Business Process Implementations Flexible and Robust: Error Handling in the AristaFlow BPM Suite. CAiSE'10 Demos, June 2010, Hammamet, Tunisia.

[6] M. Reichert, B. Weber. Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Berlin-Heidelberg. 2012.

[7] J. M. Hu, S. S. Zhang, X.Y. Yu. A Workflow Model Based on ECA Rules and Activity Decomposition [J]. Journal of Software, 2002 13(4):0761-0767. (in Chinese).

[8] Gang Ye, Xianjun Li, Dan Yu, Zhongwen Li, Jie Yin .The Design and Implementation of Workflow Engine for Spacecraft Automatic Testing JOURNAL OF COMPUTERS , VOL. 6, NO. 6, JUNE 2011 1145-1151

[9] Chaokun Yan, Huimin Luo, Zhigang Hu, Xi Li, Yanping Zhang Vol 8, No 4 (2013)842-850 Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid Abstract PDF

[10] H. Schonenberg, R. Mans, N. Russell et al. Process flexibility a survey of contemporary approaches[A]. CIAO! / EOMAS 2008: 16-30 .

[11] J. J. Li, W. P. Wang, F. Yang. Review on approaches of flexible workflow [J]. Computer Integrated Manufacturing Systems 2010 16(8):1569-1578. (In Chinese).

[12] Y. B. Guo. Research on transitional process model and technologies [D]. Ph. D. Thesis South China University of Technology. 2007 12. (in Chinese).

[13] J. Cardoso, J. Mendling, G. Neumann, and H.A. Reijers. A Discourse on Complexity of Process Models. BPM 2006 Workshops, LNCS 4103, pp. 117–128, 2006. Springer-Verlag Berlin Heidelberg 2006.

Yubin Guo (1973-) Received Ph. D. from South China University of Technology in 2007. She is now a lecturer in South China Agricultural University. Her research interests include Database theory and technology, Workflow technology, cryptography and network computing.

Zeye Cai (1990-) received bachelor degree in South China Agricultural University in 2013. He is now a master student in South China University of Technology. His research interests include Database theory and technology and Workflow technology.

Zewei Lin (1990-) received bachelor degree in South China Agricultural University in 2013. His research interests include Database theory and technology and Workflow technology.

Ximing Li (1973-) Received Ph.D. degree from College of Informatics, South China Agricultural University, Guangzhou, Guangdong, China, in 2011. He is now a PostDoc student in epartment of Computer Science, Ben Gurion University of the Negev. His current research interests include computer theory and cryptography.