Short Paper

A Workflow Event Logging Mechanism and Its Implications on Quality of Workflows *

MINJAE PARK AND KWANHOOON KIM
Department of Computer Science
Kyonggi University
Gyeonggi-Do, 443-760 Korea
E-mail: kwang@kgu.ac.kr

As the workflow/BPM systems and their applications are prevailing in the wide and variety industries, we can easily predict not only that very large-scale workflow systems (VLSW) become more prevalent and much more needed in the markets, but also that the quality of workflow (QOW) and its related topics be issued in the near future. Particularly, in the QOW issues such as workflow knowledges/intelligence, workflow validations, workflow verifications, workflow mining and workflow rediscovery problems, the toughest challenging and the most impeccable issue is the workflow knowledge mining and discovery problems that are based upon workflow enactment event history information logged by workflow engines equipped with a certain logging mechanism. Therefore, having an efficient event logging mechanism is the most valuable as well as $A$ and $\Omega$ of those QOW issues and solutions. In this paper, we propose a workflow enactment event logging mechanism supporting three types of event log information – workcase event type, activity event type and workitem event type, and describe the implementation details of the mechanism so as to be embedded into the e-Chautauqua system that has been recently developed by the CTRL research group as a very large scale workflow management system. Finally, we summarize the implications of the mechanism and its log information on workflow knowledge mining and discovery techniques.

Keywords: workflow event logging mechanism, workflow knowledge mining and discovery, quality of workflow, workflow event log format, very large scale workflow architecture

1. INTRODUCTION

In fact, workflow design and automation technologies are becoming one of the hottest technologies in the enterprize information technology arena, which means that workflow systems have been widely adopted by many organizations with not only the belief that they enable large organizations to improve dramatically the way they operate, but also the evidence that their effectiveness has also come under our observation in numerous deployments. Consequently, according for those workflow design and automation technologies to swiftly grow and be increasingly used by both traditional and newly-formed
web-based enterprises, we need to deal with and attempt to analyze a new and advanced type of requirements and demands concerning workflow intelligence and quality in terms of not only the design-time workflow verification and validation issues but also the runtime workflow execution issues. That is, the newly emerging requirement in recent has been set-up as a new toughest challenge – Quality of Workflow (QOW) [12].

Furthermore, in order to improve the quality of workflows with a high, consistent, and predictable fashion, especially it is never enough to emphasize on the workflow traceability and rediscoverability [15, 17-19, 22]: the workflows should be correctly designed; their execution should be met their workload requirements; the workflows reasonably reflect their real-world business processes; and the (human or automated) process resources should be able to perform the work items in a timely fashion. But, many of the conventional workflow products have been experienced with being ineffective for the serious and large scale workflow applications, because they provide just the bare necessities of functionality – modeling capability, scripting tools, dynamic change supports [20], and many other basic features of the workflow systems. Particularly, many successful large scale workflow customers found out that as they tried to scale up from pilot test mode to enterprise-wide mode, there were severe symptoms of inflexibility, non-recoverability, and non-verifiability [10-12] That is, it is necessary for the systems to be equipped with advanced features to effectively trace and observe the runtime behaviors of workflow procedures. The event logging mechanism ought to be the right solution for the advanced features like the workflows’ runtime behaviors trace and observation, and the workflow quality improvement [12].

Based upon these practical backgrounds and motivations, this paper conceives a workflow enactment event logging mechanism and implants it into the e-Chautauqua workflow management system [10] that has recently developed by the author’s research group. Especially, in the mechanism, the workflow enactment events log information [5, 6] being stored by the system’s engine components is classified into three types according to the corresponding engine components’ logical levels. We identified all of the types of events possibly happened in the system, and classified them based upon the statuses of workflow instances running on and being managed by the system. Through the consecutive three sections, we introduce the overall description of the e-Chautauqua workflow management system [10], and describe the details of the workflow enactment event logging mechanism and its log information types. Finally, we summarize the implications of the mechanism by showing how the log information is used in workflow mining and re-discovery frameworks.

2. THE OVERALL SYSTEM ARCHITECTURE OF E-CHAUTAUQUA

The authors’ research group has recently completed the development of a workflow management system that aims for very large scale workflows applications, and it is named e-Chautauqua workflow management system [10]. e-Chautauqua is based on the workcase-oriented workflow architecture [20], and especially we have implemented it by the Enterprize Java Beans framework approach, while almost all conventional workflow systems are based on the activity-oriented workflow architecture proposed by OMG [4, 7-9]. This section shortly introduces e-Chautauqua’s overall system architecture as an underlin-
ing workflow system of the proposed workflow event logging mechanism to be described in the remaining sections.

As shown in Fig. 1, the functional descriptions of e-Chautauqua’s architectural components and the ways they cooperate with each other are illustrated. Basically, e-Chautauqua is fully implemented so as to provide almost all services and facilities specified in the international workflow standards from the interfaces 1 through 5 announced by WFMC\(^1\). For examples, in terms of the workflow definition language, it completely handles WPDL\(^2\) as well as XPDL\(^3\), and also it is able to support the Wf-XML 2.0\(^4\) interoperability interface, which has recently released by WFMC, for interacting with other heterogeneous workflow systems. Our focus is on the relationship between the workflow enactment component (so-called workflow engine) and the log agent in charge of event logging functionality. In this section, we won’t describe the details of the e-Chautauqua’s architectural components. However, the essential components related to the logging functionality are minutely described in the next section.

![Fig. 1. The overall system architecture of e-Chautauqua.](image)

### 3. WORKFLOW EVENT LOGGING MECHANISM

As a functional part of the e-Chautauqua workflow management system, we have developed an workflow enactment event logging mechanism. In this section, we describe the functional structure of the agents directly related to the mechanism, and explain about how the engine components take events, generate the events’ log information, and finally store them on log database. Finally, we also introduce the asynchronous logging message queue mechanism that is used for the engine components to store their event log information.

---

2 Workflow Process Definition Language.
3 XML-based Process Definition Language.
4 WFMC’s Newly Released Standard supporting XML-based workflow interoperations between heterogeneous workflow engines.
3.1 Engine’s Logging Components

The overall package structure of e-Chautauqua, as depicted in Fig. 2, consists of several agent-based components, such as PDL (process definition language) Agent, OM (organization management) Agent, Administration Agent, Tool Agent, requester, worklist handler, script interpreter, Log Agent, workflow Message Beans using JMS message queue, and Enactment & Control Agent consisting of workcase objects registered in the workcase pool. These all components are deployed on an Enterprize Java Beans middleware platform\(^5\) and they asynchronously communicate with each other through a JMS-based message queue mechanism named the workflow queue. Consequentially, the proposed event logging mechanism is directly related with the log agent, requester, worklist handler and the workcase-based enactment & control agent. The log agent receives all types of event log information from the related agents and stores them formatted in the proposed XML schema. In the next section we minutely describe the details of the mechanism.

The core engine components that distinguish e-Chautauqua’s engine from other workflow engines are the workcase object components. That is, as stated in the previous section, almost all conventional workflow engines are based on the activity-oriented workflow architectural style \([13]\), but on the other hand the e-Chautauqua’s engine is based on the workcase-oriented workflow architectural style \([13]\). In the activity-oriented workflow engines, the control flow management functionality is performed through interactions among activity objects, because the core component is the activity object component. In contrast to this, the control flow management is done by the workcase objects in the workcase-oriented workflow engine (e-Chautauqua), and so the activity precedence information is stored to the inside of each workcase object as data. Conclusively, the most valuable benefit that we expect from the workcase object component is that the control flow management functionality can be done by much simpler mechanism, and also it can be efficiently done with much smaller amount of computing resources, because the workcase-oriented workflow engine is able to dramatically decrease the number of objects resided and managed in the system. These facts mean also that the workcase-oriented workflow engine like

\(^5\) Note that we used the Weblogic 8.1.2 package, but it won’t be dependent on a specific EJB package. Recently we have successfully deployed our system on JBoss enterprize middleware platform, too.
e-Chautauqua must be more appropriate and durable for the very large scale workflows, and it must give satisfaction in the higher degree of workflow instantiation complexity.

3.2 The Implementation Details

Fig. 3 is to present the relationship and interactions between the e-Chautauqua engine’s workcase components and the log agent components that are in charge of the execution of the event logging mechanism proposed in this paper. As shown in the figure, once one of the workflow clients requests its assigned activity enactment services, then its corresponding workcase component performs the requested services, and makes their corresponding events log information according to the XML-based log message format to be specified in the next section and stores to log databases through the corresponding log agent. The detailed role and functionality of the log agent is described in the next section.

Fig. 3. e-Chautauqua’s events logging mechanism.

All events coming from the e-Chautauqua workflow enactment components, such as worklist handler, requester and workcases, would have to be logged on a certain type of storage. These events log information is precisely well-defined in the audit and monitoring functions’ standard specifications of WfMC [2, 3]. However, these audit and monitoring information have not defined by any types of XML-based formats. So, in this section, we try to identify and classify all events produced by the engine components, and also to define them by a certain form of XML-based representation formats.

Structure of the Mechanism The functional structure of the workflow event logging mechanism is depicted in Fig. 4. It consists of the following three types of components:

- Event triggering components: Requester and Worklist Handler
- Event formatting components: Workcase Pool
- Event logging components: Log Agent and Log File Storage
The event triggering components handle the workflow enactment services requested from workflow clients, and, these services are able to be categorized into three levels of classification – Workcase level class, Running activity level class, and Workitem level class. The event formatting components try to compose event log messages according to the service classes after performing the requested services. Finally, the event logging components, especially the log agents, take in charge of the responsibility of the event logging mechanism. Once, a log agent receives event logs and then transforms them into XML-based log messages, and store the transformed messages onto the Log File Storage.

Workflow Event Log Information  As shortly explained in the previous section, the workcase components, which are taking a role of the event formatting component, compose event log messages after executing the requested services from the event triggering components – the requester and the worklist handler. After doing the formatting job, they transmit the formatted event log messages to the event logging components – the log agents. Based on the formatted messages, the log agents form the XML-based event log information. In order to efficiently perform these logging-related jobs, we classify the events into three levels of classes – workcase level event class, running activity level event class, and workitem level event class. These event classes’ message formats are precisely described in Figs. 5, 6, and 7, respectively. As shown in the EventCode rows of the tables, the detailed names of the events that are captured and logged by the mechanism are summarized as the following:

- Workcase Level Events: WMCreatedWorkcase, WMStartedWorkcase, WMChangedWorkcaseState, WMCompletedWorkcase, WMTerminatedWorkcase, WMAbortedWorkcase
- Running Activity Level Events: WMChangedActivityInstanceState, WMCompletedActivityInstance, WMTerminatedActivityInstance, WMAbortedActivityInstance
- Workitem Level Events: WMAssignedWorkitem, WMGetWorkitem, WMChangedWorkitemState, WMCompletedWorkitem

Operation of the Mechanism  Finally, the components related to the event logging
Fig. 5. Workcase level event log Information.

Fig. 6. Runtime activity level event log information.

Fig. 7. Workitem level event log information.
Fig. 8. e-Chautauqua’s XML-based event log samples.

mechanism communicate with each other through the asynchronous logging message queue mechanism that uses a message-driven bean for realizing the reliable message transmission and that is based upon the JMS (Java Message Server) queue as an asynchronous message communication channel. That is, a component sending a log message composes the message in one of the event log message formats and transmits it to a JMS queue, and then the JMS queue looks up the message map to find out the target component and its method and parameters by interpreting the received message from the sending component, and finally the JMS queue invokes the method for passing the parameters’ values or for returning the execution results. The figure shows an example issued by one of the event triggering components, in which the sending component is a runtime client trying to request starting a workcase, and the target component is the requester that takes in charge of the responsibility for starting the workcase.

Based upon the workflow event log classes and their messaging queue mechanism, we are able to realize an XML-based event logging format and its delivery to the log agents. Fig. 8 is to show three samples of the event log classes captured from the real operations. Especially, we would emphasize that it was possible to be able to show you the samples in the figure, because we implemented the proposed logging mechanism as embedded components of e-Chautauqua workflow management system.

4. IMPLICATIONS ON QUALITY OF WORKFLOWS

The workflow enactment event log information generated by the proposed logging
A workflow event logging mechanism [14] has very important implications on performing a series of advanced techniques not only for improving the quality of workflows but also for discovering useful knowledge. The successful implementation of the logging mechanism ought to have very influential implications on the issue of enhancing the quality of workflows. Consequently the following crucial functionalities for achieving the quality of workflows can be possibly realized:

- Workflow runtime status monitoring and statistical reporting functionality
- Workflow runtime recovery functionality
- Workflow knowledge mining and discovery functionality
- Workflow validation functionality
- Workflow process mining and rediscovery functionality

Based upon the logging mechanism and its event log classes, we have developed an XML-based workflow event log language, which is abbreviated to XWELL [24]. Also, it is possible to naturally organize a feasible framework coping with the workflow quality issue through the XML-based log information as shown in Fig. 9. The framework consists
of modeling part and mining and discovery part for the quality of workflows. The workflow modeling part (the upper part of the figure) takes charge of modeling and analyzing workflow models. The mining and rediscovery part (the lower part of the figure) is of re-discovering and/or mining the enacted workflow models from the workflow warehouse built from the XML-based workflow log information (formatted in XWELL) that is generated from the proposed event logging mechanisms resided in the log agents of workflow engines. The workflow engines must be no matter whether they may homogeneous or heterogeneous if they embed a certain type of XML-based event logging mechanism handling with the XML-based event log message format in XWELL [24].

Finally, the followings are to shortly introduce the research and development fields that may be implied with the workflow event logging mechanism proposed in the paper:

**Workflow Process Rediscovery**  Basically workflow management systems help to execute, monitor and manage work process flow and execution. These systems, as they are executing, keep a record of who does what and when (e.g. log of events). The activity of using computer software to examine these records, and deriving various structural data results is called workflow mining [19] or workflow process rediscovery [22]. The workflow process rediscovery activity, in general, needs to encompass behavioral (process/control-flow), social, informational (data-flow), and organizational perspectives as well as other perspectives [21], because workflow systems are “people systems” that must be designed, deployed, and understood within their social and organizational contexts. Particularly, in terms of rediscovering the behavioral perspective of a workflow model, we would introduce its basic principle.

A workflow model is described by several entities, such as activity, role, actor, invoked applications, and relevant data. The control flow perspective specifies the transition precedences-sequential, conjunctive (AND), disjunctive (OR) and loop execution sequences-among the activities. In other words, the event log is an interleaved list of events from executing numerous workcases-workflow instances of a workflow model. By examining the log, we can detect the temporal ordering of activity executions for each workcase, and then infer their control flow perspective patterns. As a simple example, suppose that we examine the log of a structured workflow process that is made up of four activities, $a_1$, $a_2$, $a_3$, and $a_4$. Suppose also that all the four activities are always executed in some order by each workcase. If we observe over a large number of workcases that $a_1$ is always executed first and $a_4$ is always executed last, then we can begin to piece them together into a workflow process model that requires $a_1$ to complete before all other activities, and $a_4$ to execute after all others. If we find workcases in the log where $a_2$ begins before $a_3$, and other cases where $a_2$ begins after $a_3$, then we can infer that the workflow process begins with $a_1$; after it completes, $a_2$ and $a_3$ execute concurrently (Conjunctive Transition: AND Control Flow); and after they both complete, then $a_4$ executes. Conclusively, the proposed workflow event logging mechanism and its formatting language are directly related with these workflow process rediscovery activities and techniques.

**Workflow Knowledge Discovery**  The workflow event logging mechanism and its formatting language like XWELL may affect to the techniques and methods of workflow knowledge discovery. Here is an example of workflow knowledge discovery issue; It is the case of reachable-path rediscovery problem [12] that gives a way to efficiently re-
discover the discrepancy between the original workflow process model as it is built and the enacted workflow process (workcase) as it is actually executed. The discrepancy, as you can easily imagine, is caused by the alternative paths exhibited on the model. The number of alternative paths on the model will effect on the degree of the discrepancy. For example, after rediscovering a workcase from workflow logs, we need to know along which reachable-path the workcase has followed. This might be very useful knowledge for workflow administrators and designers to redesign and re-estimate the original workflow process model after being elapsed a specific amount of period. Conclusively, this statistical runtime information should be very effective and valuable knowledge for redesigning and reengineering the workflow model, and the proposed mechanism ought to be implied with this workflow knowledge discovery issue, too.

Social Network Discovery on Workflow  The workflow event logging mechanism and its formatting language have something to do with those algorithms for discovering a work-sharing human network embedded on a workflow model, and also for quantifying the degree of working-intimacy among humans through the essential notions of the centrality-degree centrality, betweenness centrality, stress centrality and closeness centrality-sharing the enactment of the workflow procedure. Basically, the recent issue of social network discovery is to define a discovered human network from the workflow enactment event logs, to evaluate the degrees of working-intimacies in the discovered human network, and finally to visualize the degrees of working-intimacies among humans through a commercialized tool used to visualizing social networks, based on the centrality analysis result over the discovered human network.

Summarily speaking, the workflow event logging mechanism described in the paper ought to be an essential component for conceiving techniques and methods resolving those issues introduced above. In the near future, we are going to make some discovery techniques of workflow knowledge and social networks based upon the logging mechanism with XWELL.

5. CONCLUSIONS

In this paper, we have newly proposed the XML-based workflow logging mechanism and introduced its implications on the quality of workflow issue. And, in order to show that the mechanism properly works in the real computing environment, we have implemented and implanted in the e-Chautauqua workflow management system that has recently implemented by the author’s research group. Particularly, the event level classes and their logging formats suggested in this paper should have to be a guideline for implementing an XML-based workflow log message formats that are possibly referred by other workflow research groups who have plans to develop such a workflow logging mechanism. In recent, the literature needs various, advanced, and specialized workflow mining techniques and rediscovery algorithms that are used for finally providing feedbacks of their analysis results in order to the redesign and reengineering phase of the existing workflow and business process models. We strongly believe that this work might be one of those impeccable attempts and pioneering contributions for those quality of workflow issues.

In this paper, summarily speaking, our emphasis is placed on the necessity of the
quality of workflow as well as the workflow intelligence that aim at improving the quality of the corresponding workflow model. Therefore, in the near future, we will make progress on the important subjects like “what is the quality of a workflow model?,” and “how can we evaluate the quality of a workflow model?”. And we try to answer for those questions through the concept of XML-based workflow event logging mechanism.

REFERENCES

132-136.

Minjae Park is a senior member of research staff of BISTel, Inc. He received B.S., M.S. and Ph.D. degrees from the Computer Science Department of Kyonggi University, South Korea, in 2004, 2006 and 2009, respectively. His research interests are workflow mining, graphical workflow modeling, and service-oriented workflow.

Kwanghoon Kim is a full Professor of Computer Science Department and Director of the Contents Convergence Software Research Center at Kyonggi University, South Korea. At Kyonggi, he is involved in research and teaching of workflow, business process management, groupware, coordination theory, computer networks, software architectures, and database systems. He received B.S. degree and M.S. degree in Computer Science from Kyonggi University and Choongang University, respectively. And, he also received his M.S. and Ph.D. degree from the Computer Science Department of the University of Colorado at Boulder, in 1994 and 1998, respectively. He had worked as a researcher and developer at Aztek Engineering, American Educational Products Inc., and IBM in USA, as well as at Electronics and Telecommunications Research Institute (ETRI) in South Korea. In present, he is a vice-chair of the BPM Korea Forum, a country-chair (Korea) and a ERC vice-chair of the Workflow Management Coalition. He has also been on the editorial board of the journal of KSII, and the committee member of the several conferences
and workshops. His research interests include workflow systems, adaptive case management, service-oriented business process management, workflow discovery/rediscovery, CSCW/Groupware, collaboration theory, Grid/P2P/Cloud distributed computing, workflow mining systems and RFID/USN middleware.