IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4594

Management of Document Versions in Workflow Systems

Yeongho Kim[†1)], Hyerim Bae[1)], Yong Tae Park[1)], and Woo Sik Yoo[2)]
[1)]Department of Industrial Engineering, Seoul National University; Seoul, 151-742, S. Korea; Tel: 82-2-880-8335, Fax: 82-2-889-8560; E-mail: yeongho@snu.ac.kr
[2)]Department of Industrial Engineering, University of Inchon; Inchon, 402-749, S. Korea; Tel: 82-32-770-8488, Fax: 82-32-770-8488; E-mail: wsyoo@incheon.ac.kr

ABSTRACT

This paper proposes a new method of document version management for workflow management systems. Recently, a workflow management system is considered as an essential element for automation of complex business processes, particularly for those in an e-Business environment. A core element of such processes is the documents that flow over the processes. Therefore, it is very important to have a systematic management of document changes along with the process execution. We propose a version model that can take into account the structure of the underlying process over existing version management techniques. In this model, the components of a document and a process are associated each other, and this becomes the basis for automatic creation of document versions and automatic configuration of relevant document for a certain user at a certain stage of process. A prototype system has been implemented, and the potential advantages of the approach have been discussed.
Keywords: Business process, Workflow, Version management, Electronic document management

1. INTRODUCTION

For the last several years, it has been conceived that WorkFlow Management System (WFMS) is an essential element for automation of complex business processes [1]. The WFMS is a software system that defines, controls and manages business processes [6], [7]. A business process usually involves documents, and in many cases, filling in the documents is considered as an important part of carrying out work in the process. Therefore, efficient handling of documents is of great significance in business process management.

A business process usually involves many participants who may deal with the same document on the process. As soon as one finishes one's task in the document, it is handed over to the next participant. This order of task sequence is specified in a process model. In a certain task, the responsible employee needs to work on a relevant version of the document, and updates it. This is repeated until the whole process is completed. In this setting, it is often very important to identify who is responsible for what part of the document contents. The importance may be doubled in e-Business environments in which many companies exchange documents.

Document management, as such, is an important part of business processes. Therefore, many commercial WFMS's provide functions dealing with document management to some extent. They, however, are limited to simple storage services and delivery of documents. A process execution usually accompanies document changes, such as adding, modifying, and deleting some of the document contents. This produces the necessity of managing document changes along with process execution. To the best of our knowledge, there is yet no system that can provide systematic management of document changes while taking into account the underlying process controlling the document flow.

To overcome the above limitation in conventional WFMS's, we propose a new version creation model. The essence of the proposed approach is at the fact that it takes into account the semantics of underlying processes. We modularize a document into several components, called workunits in this paper, and associate each workunit with the activities that are supposed to handle it. We also propose a run-time model with which document versions can be created during process execution.

2. WORKFLOW AND VERSION MANAGEMENT

Workflow management is a term for a diverse and rich technology to support business process automation. In almost all WFMS's, defining a process model is prerequisite to automatic execution and control of the actual process. A process model is a coordinated set of activities, and an activity is a logical step or description of a piece of work. A WFMS first specifies a workflow process by defining activities that contribute to achieving the business objectives intended by the process and establishing the relations among the activities. While the process is being executed, the resources, like documents or application programs that are needed to perform each activity, are delivered automatically by the WFMS.

Version management, in its broadest sense, is a systematic method of dealing with changes of objects over time, and version is defined as a snapshot of an object that is semantically meaningful at a point in time [8]. The object changes are usually represented in a graphical form, and this is called version graph [8]. There are two different types of versions that are *revision* and *variant* [3]. Revision is a relation between two versions that are directly interlinked in a version graph. This is established when one of them is created by modifying the other. On the other hand, variant is the relation defined over two or more versions that are generated independently from the same previous version. This relation is not explicitly indicated in a version graph, but appears as a set of parallel paths.

There has been much research work in the field of version models [3], [8], and the version management has been successfully applied to such areas as software configuration management [3], engineering data management [5], [8], and temporal database. Although the application areas are different, the models used are similar to each other in that they manage the change of objects over the passage of time.

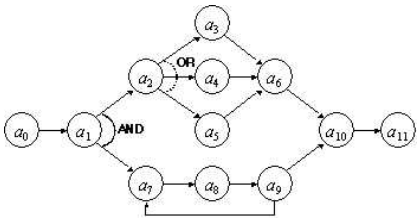
3. WORKFLOW-BASED VERSION MANAGEMENT

3.1. Process and Document Structure Models

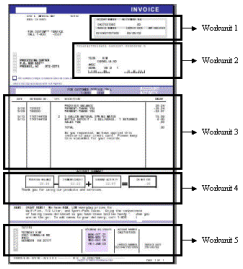
A typical workflow system runs on a process model, like the example in Figure 1 (a). It, in general, represents activities, their relations, and attributes describing the process and activities. We can classify the process flow into serial, AND-parallel, and OR-parallel. A serial process is one that does not involve any split and merge, whereas an AND-parallel and OR-parallel process has split and merge. The latter two process types include more than one branches between the split and merge activities. Notice that a workflow process can be modeled with a combination of those types [7]. There is a huge body of literature in the process model, and readers can refer to [2], [10].

Business documents are usually very well structured, and the format is predetermined. Such a document is called form document. We partition a form document into a set of logical parts, each of which becomes a unit of work dealt with by an activity. The unit of work is called a 'workunit' in this paper.

Figure 1. Examples of process model and form document



(a) Process model



(b) Form document

Figure 1 (b) shows an example form document and its workunits. Once a process model and a document model are prepared, each workunit is assigned to some activities. The relation defined on the pair of workunit and activity is called assignment association. Every workunit has to be assigned to at least one activity, whereas an activity can have nothing to do with any workunit or deal with multiple workunits.

3.2. Document-Workunit Relations

In addition to the build-time models of process and document structures in the previous section, we need a run-time model that consists of version graphs and document-workunit relations. A version graph records the history of changes of an object that can be either a document or a workunit.

Creation of a new workunit version always leads to forming a new document version. In addition, a certain version of document can actually be considered as a collection of workunit versions. Document-Workunit Relation (DWR) represents such relations between a document version and a workunit version as follows.

Definition 1 (Document-workunit relations)

Consider a document d and its workunit w. The *DWR* relation, established between the p-th version of d and the q-th version of w, i.e., $v_p(d)$ and $v_q(w)$, is one of the following three types.

- The relation, $v_p(d) \leftrightarrow^I v_q(w)$, is an initialization relation (DWR^I) indicating that both $v_p(d)$ and $v_q(w)$ are the initial versions, that is, $p = q = 0$.
- The relation, $v_p(d) \leftarrow^G v_q(w)$, is a generation relation (DWR^G) stating that the workunit version, $v_q(w)$, generates the document version, $v_p(d)$.
- The relation, $v_p(d)$ ®^{C} $v_q(w)$, is a composition relation (DWR^C) expressing that the workunit version, $v_q(w)$, is an element of the document version, $v_p(d)$.

4. TYPES OF VERSIONS

Based on the build- and run-time models described in the previous sections, we are able to manage versions while executing business processes. Since the operations of version creation are different depending on the types of process flow, they need to be described for each of the types. Due to the space limitation, a brief introduction to each of the type is presented below. A more detailed explanation is available in [3].

4.1. Serial Process

Activities in a serial process are all linearly connected. Such a serial process generates versions having revision relations. Consider the serial process in Figure 2 (a). The process deals with document d, and the document's workunits are assigned to the activities as indicated in the figure. The document that has to be checked out in a serial process is always the latest version. The check-out procedure identifies the workunits that the latest document version consists of, and constructs it by simply putting them together. On the other hand, check-in procedure simple adds up a revision to the current version graph.

4.2. AND-parallel Process

An AND-parallel process allows multiple activities to be processed simultaneously. The split activities are independent of each other, and the workunits assigned to the activities also need to be dealt with in parallel. In this paper, AND-parallel processes are further classified into competitive split, cooperative split, and combined split. Depending on these split types, document versions are managed differently.

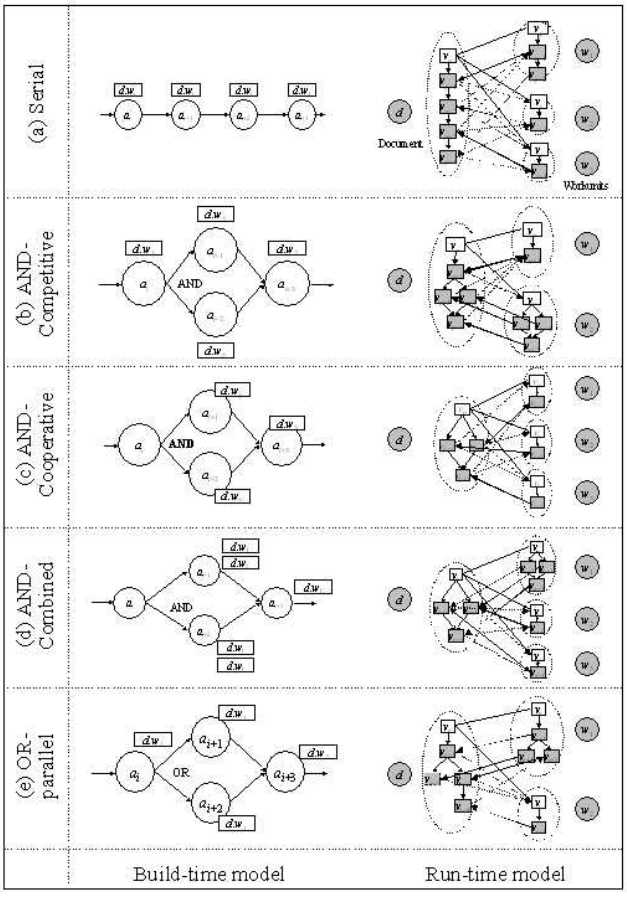
Competitive split

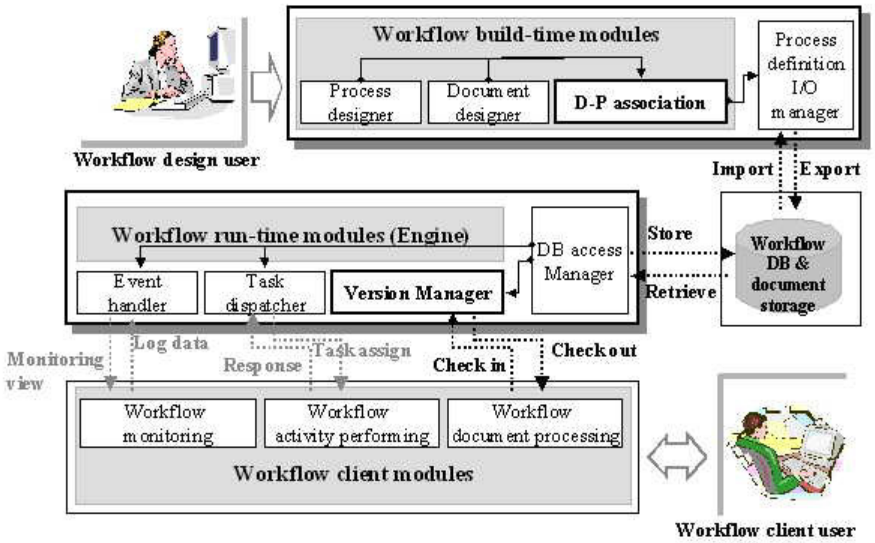
A competitive split is an AND-parallel process such that the same workunit is assigned to every branch of the split process. That is, after a process splits, all the branches check out the same document and work on the same workunit, but each of them generates its own version. Therefore, a competitive split produces several alternatives for one workunit. However, not all the alternatives are meaningful in the succeeding process. It can be considered that the branches compete to produce an alternative that is used in the final version. This is the reason we call it a competitive split. The winning version could be automatically determined if the business logic is well understood and thus it can be codified. Otherwise, it is manually chosen. A simple example of competitive split is presented in Figure 2 (b).

Cooperative split

A cooperative split is a type of parallel process where each branch deals with different workunits. It is assumed that dependency doesn't exist among the workunits so that they can be processed in parallel. After finishing all the branches, the collection of the resulting workunits can form a document version. It can be seen that all the branches cooperate together to produce one

Figure 2. Process types and version creation models





document version. This explains the name of the split. A simple example of cooperative split is shown in Figure 2 (c).

Combined split

A combined split is a combination of competitive split and cooperative split. Sometimes, the assignment of workunits to process activities cannot be explained by only one of the two split types. That is, some workunits are in a competitive split while the others a cooperative split. In such a case, the combined split is used. An example process of a combined split is presented in Figure 2 (d). Notice that every branch deals with workunit w_1. On the other hand, w_2 and w_3 are processed in different branches. While executing a combined split, the versions for w_1 is created based on the competitive split, and those for w_2 and w_3 follow the cooperative split.

4.3. OR-parallel Process

An OR-parallel process is similar to an AND-parallel process in terms of its process structure. It has split and merge activities and branches. However, not all the branches are meaningful at run-time. Some of the branches are selected and activated, and the process terminates when one of them finishes successfully. The branches that have not finished yet are simply canceled. At build-time, it is impossible to know which branch reaches to completion. Henceforth, in our version model, each branch maintains a version graph, but the one for successful branch becomes effective after merging the branches. An example of OR-parallel process is shown in Figure 2 (e).

5. PROTOTYPE IMPLEMENTATION

We have implemented a prototype system for the models proposed in this paper. The system is implemented on top of an existing WFMS, called SNUFlow [9], by adding component modules that provide version management functions. Some of the functions can be accessed at 'http://workflow.snu.ac.kr:8080/SNUFlow/client.jsp'.

The overall system architecture is presented in Figure 3. The system includes build-time, run-time, and client modules. The build-time modules include a process designer, document designer, and Document-Process (D-P) association component. The process designer and document designer provide interfaces for designing processes and documents and the D-P association component for establishing relations between processes and documents. These altogether describe how a process will be automatically executed at run-time. The specifications are imported from or exported to a workflow storage via an I/O manager.

In order for a user to easily carry out an activity, the prototype system identifies and delivers a right version of document to the user, so that the user can readily check out the document assigned to the activity. The user can simply click a mouse to check out the document version. On completion of the activity, the user can check in the new version of the document, then the system automatically updates the versions of workunit and document.

6. SUMMARY AND CONCLUSIONS

The main purpose of our research is to develop a method of managing changes of documents in workflow processes. The essence of the proposed approach is that it takes into account the semantics of underlying processes. Our approach provides several advantages as follows. First, it helps workflow users by automatically checking out the right document version that the users have to work. Second, users can have a better understanding of the document changes. This is because the document changes are tightly associated with the underlying workflow processes, and our system can visualize it. Third, it is possible to recover a document into an earlier version that has been created before. When the process execution needs to be returned to a previous activity due to system errors or exceptional cases, it is required to recover the document at that activity. We think our version model can be a solution to the issue of workflow recovery. Fourth, since it is now possible to readily identify the content changes associated with a certain version creation, the approach would increase the responsibility of the employee in charge of the version creation.

An interesting further research issue is to support cooperative authoring in computer supported collaborative work environments. A cooperative authoring process involves multiple authors and thus many changes may take place even at the same time. It is important but not an easy task to support systematic versioning in the environment. Another issue is to develop standardized API's or standard versioning protocol for version management to interface with different WFMSs.

ACKNOWLEDGEMENT

This research was partly supported by the program of National Research Laboratory granted from Korea Institute Science and Technology Evaluation and Planning. It was also supported by the Korea Science and Engineering Foundation (KOSEF) through the Northeast Asian e-Logistics Research Center at University of Incheon.

REFERENCES

[1] W. M. P. van der Aalst, Process-oriented architecture for electronic commerce and interorganizational workflow, Information Systems 24 (8) (1999) 639-671.

[2] W. M. P. van der Aalst and A. H. M. ter Hofstede, Verification of workflow task structures: A petri-net-based approach, Information Systems 25 (1) (2000) 43-69.

[3] H. Bae, W. Hur, W. S. Yoo, and Y. Kim, "Document Versioning on Workflow Processes," Submitted to Computers in Industry, 2002.

[4] R. Conradi and B. Westfechtel, Version models for software configuration management, ACM Computing Survey 30 (2) (1998) 232-282.

[5] K. R. Dittrich and R. A. Lori, Version support for engineering database systems, IEEE Transactions on Software Engineering 14 (4) (1988) 429-437.

[6] D. Georgakopoulos, M. Hornick, and A. Sheth, An overview of workflow management: from process modeling to workflow automation infrastructure, Distributed and Parallel Databases 3 (1995) 119-153 (also available at http://citeseer.nj.nec.com/georgakopoulos95overview.html).

[7] D. Hollingsworth, Workflow management coalition specification: The workflow reference model, WfMC specification, WFMC-TC-1003, http://www.wfmc.org, 1995.

[8] R. H. Katz, Toward a unified framework for version modeling in engineering database, ACM Computing Surveys 22 (4) (1990) 375-408.

[9] Y. Kim, S. Kang, D. Kim, J. Bae, and K. Ju, WW-Flow: Web-based workflow management with runtime encapsulation, IEEE Internet Computing 4 (3) (2000) 55-64.

[10] G. Mentzas, C. Halaris, and S. Kavadias, Modelling business process with workflow systems: An evaluation of alternative approaches, International Journal of Information Management 21 (2) (2001) 123-135.

[11] E. Sciore, Versioning and configuration management in an object-oriented data model, VLDB Journal 3 (1994) 77-106.