A Version Management of Business Process Models in BPMS

Hyerim Bae[1], Eunmi Cho[1], and Joonsoo Bae[2,*]

[1] Department of Industrial Engineering
Pusan National University, San 30, Jangjeon-dong, Geumjeong-gu,
Busan, Korea, 609-735
hrbae@pusan.ac.kr, oldlace@naver.com
[2] Department of Industrial and Information Systems Engineering
Chonbuk National University, 664-14, Dukjin-dong, Duckjin-gu,
Jeonju, Korea, 561-756
jsbae@chonbuk.ac.kr

Abstract. BPM system manages increasing number of business processes and the necessity of managing processes during whole process lifecycle from process modeling to process archiving has been emerged. Despite of wide use of the BPM system and the maturing of its technology, the main focus has been mainly on correctly executing process models, and convenient modeling of business processes has not been considered. In this paper, a new method of versioning business processes is developed in order to provide users with an easy modeling interface. Version management of a process enables a history of the process model to be recorded systematically. In our method, an initial version and changes are stored for each process model, and any version can be reconstructed using them. We expect that our method enhances the convenience of process modeling in an environment of huge number of business processes, and thereby assists the process designer. In order to verify the effectiveness of our method, a prototype system is presented in this paper.

Keywords: Business Process Management, Version Management, XML.

1 Introduction

A business process is represented as a flow of tasks, which are either internal or external to the enterprise. Business Process Management (BPM) is an integrated method of managing processes through their entire lifecycle. The BPM system is a software system that models, defines, controls and manages business processes [6, 7]. By contrast with the simple, linear versioning methods of existing systems, the version management method presented in this paper allows parallel versioning, automatic detecting of changes and the option of keeping track of the change history

* Corresponding author.

K.C. Chang et al. (Eds.): APWeb/WAIM 2007 Ws, LNCS 4537, pp. 534–539, 2007.

with a graphical tool. With its exactingly developed functions, our method also improves user convenience and minimizes space used to store process models. Version management, in its broad sense, is a method of systematically handling with temporal features and changes in objects over time. A version is defined as a semantically meaningful snapshot of an object at a point in time [5]. A version model is used to represent a history of changes to an object over time, or simply records a version history of that object. The history of change is usually described using a version graph, such as that shown in Figure 1. In a version graph, a node is a version, and a link between two neighboring nodes is a relationship between them. For example, version $v_2(o)$ was created by modifying a previous version, $v_1(o)$.

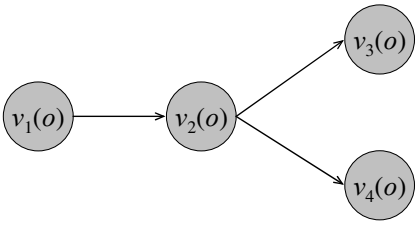

Fig. 1. Version Graph

2 Business Process Model

In this chapter, we define the process model, which is a target object of our version management. A business process model used in the BPM system is usually composed of basic objects such as tasks, links and attributes[8]. Attributes describe features of the objects. We define objects as the elements of a process model.

Definition 1. ***Business Process Model***
A process model p, which is an element of a process model set P, consists of tasks, links, and attributes. That is, $p = (T, L, A)$.

- A set of tasks: $T = \{t_i \mid i = 1,\ldots, I\}$, where, t_i represents i-th task and I is the total number of tasks in p.
- A set of links: $L = \{l_k = (t_i, t_j) \mid t_i, t_j \in T, i \neq j\}$, where, l_k represents a link between two tasks, t_i and t_j. A link represents a precedence relation between the two tasks. That is, the link (t_i, t_j) indicates that t_i immediately precedes t_j.
- A set of attributes: A is a set of task attributes or link attributes.
  1) $A_i = \{t_i.a_s \mid s = 1,\ldots, S_i\}$ is a set of task attributes, where $t_i.a_s$ represents s-th attribute of task t_i and S_i is the total number of t_i's attributes.
  2) $A_k = \{l_k.a_s \mid s = 1,\ldots, S_k\}$ is a set of link attributes,where $l_k.a_s$ represents s-th attribute of link l_k and S_k is the total number of l_k's attributes.

3 Version Management of Business Process

This chapter explains our method of process version management. We first define a version graph by introducing concepts of object, a version, and change types. Then, we present a version management algorithm using check-in/check-out algorithms.

3.1 Version Graph

In the BPM system, process design procedure is a work of defining all the process elements introduced in Section 2 by using a design tool. In this procedure, it may be impossible to prepare a perfect process model at once. Therefore, business requirements for reusing the previous models have been always raised. A user can modify a previous model to make it more complete one. After the user modifies a process model, change of the process is defined as a set of changes to component objects. A component object o can be a task, a link or an attribute. That is, $o \in T$, $o \in L$ or, $o \in A$.

A process version results from user's modifications in designing a process model. Versions are recorded using a version graph. In the version graph, a node represents a version of a process p, which is denoted as $v(p)$. A modification of a process includes changes to the objects in it, and each of the changes is represented using δ. We define a process modification as a set of object changes $\Delta = \{\delta_q(o) \mid q=1,\ldots, Q$ and $o \in T, L, A\}$. Applying the changes to a previous version to create a new version is represented by '$\cdot$' (change operation). If the n-th version of p is derived from the m-th version of p by applying the changes Δ_{mn}, we represent that $v_n(p)$ is equal to $v_m(p) \cdot \Delta_{mn}$. A version graph is defined as follows.

Definition 2. ***Version Graph, VG***
A version graph is used to record the history of a single process. Let p denote a process, and $v_m(p)$ the m-th version of the process. A version graph (VG) of p is a directed acyclic graph $VG = (V, E)$, where V and E is a set of nodes and arc respectively.

- $V = \{v_m(p) \mid m= 1, \ldots, M\}$
- $E = \{(v_m(p), v_n(p)) \mid v_m(p) \in V, v_n(p) \in V, v_n(p) = v_m(p) \cdot \Delta_{mn}\}$

In a version graph, if a version $v_n(p)$ can be derived from $v_m(p)$ by modifying $v_m(p)$ repetitively($\{(v_m(p), v_k(p)), (v_k(p), v_{k+1}(p)), \ldots, ((v_{k+l}(p), v_n(p))\} \subset E$), we say that $v_n(p)$ is 'reachable' from $v_m(p)$.

3.2 Combination of Changes

In general, a change can be classified into three types: adding, modifying or deleting. In Figure 1, we consider a component object o, which is $t_{order}.a_{due}$. If a value is added to the object, and then the value is modified into another value, the change can be represented as follows.

$$\delta_1(o) = ADD\ t_{order}.a_{due}(\text{"2005-12-24"})$$
$$\delta_2(o) = MOD\ t_{order}.a_{due}(\text{"2005-12-24"}, \text{"2006-01-24"})$$

While designing a process, it is usual that the process is modified repetitively, and multiple changes are created. These multiple and repetitive changes for the same object can be represented by a single change. For example, the two changes δ_1 and δ_2 in the above can be combined using a combination operator, '$\circ$'.

$$\delta_{new} = \delta_1(o) \circ \delta_2(o) = ADD\ t_{order}.a_{due}\ (\text{“2006-01-24”})$$

The combination of changes is calculated by using our rules, which are summarized in Table 1. Reverse change (δ^{-1}) is used for δ and empty change ($\delta_{\emptyset}$) is used for the rules. Based on the combination of object changes, we can extend and apply to combinations between sets of changes. A combination of two change sets Δ_1, Δ_2 ($\Delta_1 \circ \Delta_2$) is defined as a set of changes. All of the object changes are included as elements of the set, and two changes from the two sets for the same object should be combined into one element.

Table 1. Combination of changes

reverse change		combined change	
ADD operation	$\delta_{ADD}^{-1} = \delta_{DEL}$	ADD and DEL operation	$\delta_{ADD} \circ \delta_{DEL} = \delta_{\emptyset}$
		ADD and MOD operation	$\delta_{ADD} \circ \delta_{MOD} = \delta_{ADD}'$
DEL operation	$\delta_{DEL}^{-1} = \delta_{ADD}$	DEL and MOD operation	$\delta_{DEL} \circ \delta_{ADD} = \delta_{MOD}$
MOD operation	$\delta_{MOD}^{-1} = \delta_{MOD}'$	MOD and MOD operation	$\delta_{MOD} \circ \delta_{MOD}' = \delta_{MOD}''$
		MOD and DEL operation	$\delta_{MOD} \circ \delta_{DEL} = \delta_{DEL}$

When we apply combination operators, the following axioms are used.

- Communicative law is not valid. ($\delta' \circ \delta'' \neq \delta'' \circ \delta'$)
- Associative law is valid. ($\delta' \circ (\delta'' \circ \delta''') = (\delta' \circ \delta'') \circ \delta'''$)
- De Morgan's law is valid. ($(\delta' \circ \delta'')^{-1} = \delta''^{-1} \circ \delta'^{-1}$)
- All changes are unaffected by empty change. ($\delta' \circ \delta_{\emptyset} = \delta_{\emptyset} \circ \delta' = \delta'$)
- Associative operation between the change and reverse change is equal to empty change. ($\delta' \circ \delta^{-1} = \delta^{-1} \circ \delta' = \delta_{\emptyset}$).

3.3 Version Management Procedure

Our version management method is based on two important procedures; check-in and check-out[2, 3]. When a user wants to make a new process model from an existing model, he can request a previous version of the process to be taken out into his private work area. This is called a ‘check-out’ procedure. After a user finishes modifying the process, he may want to store it in a process repository as a new process version. We call this a ‘check-in’ procedure. That is, check-out transfers a process model from public storage to an individual workplace, and check-in returns the model to the public storage.

If all of the versions of a process are stored whenever a new version is created, storage space might be wasted. To avoid such a waste of space, we use the modified delta method [4]. The modified delta method is implemented using our check-in/out algorithms. The modified delta method uses the combination operators. It stores only a root version of a process and changes, and reconstructs any version when a user retrieves that version.

Check-out is invoked when a user requests a certain version of a process, $v_m(p)$. It first searches the changes from database, which are required to reconstruct the version and combines them into a single change. Then, by applying the change to a root version $v_0(p)$, the requested version can be reconstructed and sent to the user.

Conversely, check-in is invoked when a user returns the modified version of a process. First, it identifies which objects were changed. Then it detects the change types (add, modify, delete). The changes are stored as a change set, and finally the process version graph is updated. In Figure 2, we provide flows of the two procedures.

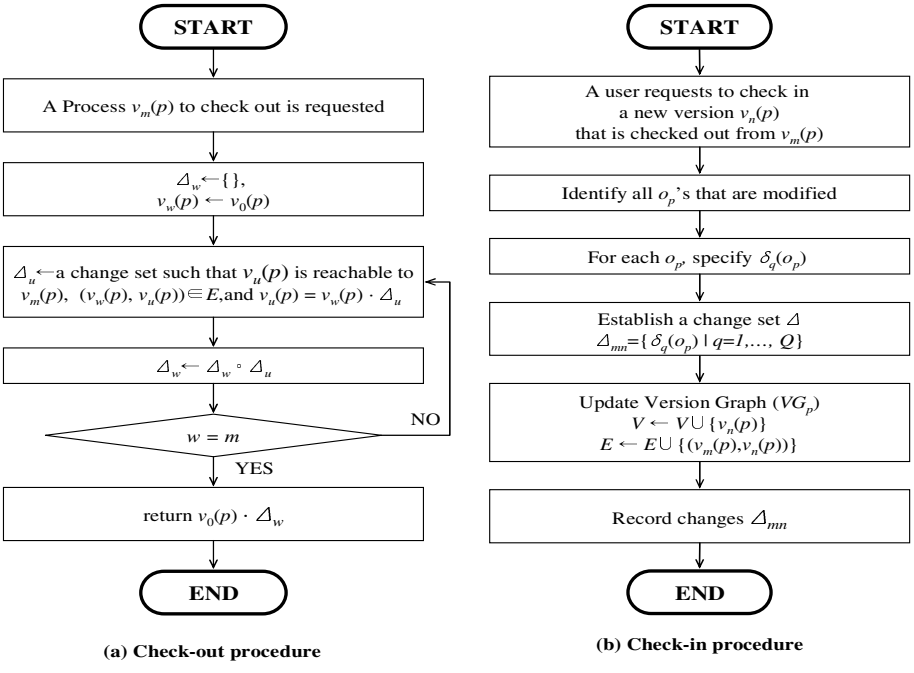

Fig. 2. Flowcharts of check-out/in procedures

3.4 Prototype System

The proposed method is implemented as a module of process designer, which implementation is a build-time function of our BPM system, called ILPMS (Integrated Logistics Process Management System) [1]. A user designs the process models with a process design tool and the designed process model is stored in a process DB. The user can easily modify and change the process using the modeling tool supported by our version management. When a user requests a process version, the system, with the check-out function, automatically generates the requested process version. After modifying the version delivered to the user, he checks in the newly created version.

4 Conclusions

In this paper, we propose a new method of process version management. Our method enables BPM users to design process models more conveniently. Though the BPM system is becoming increasingly essential to business information system, the difficulty of process modeling is a significant obstacle to employing the system. Consequently, beginners have not been able to easily design business processes using the BPM design tool. For this reason, we presented process models that use XML technology. If a user modifies a process model, our system detects the changes in the XML process definition. Then, the changes are recorded and the version graph is updated. With the version graph, we can manage history of process model change systematically. Any version of a process can be reconstructed, once its retrieval has been requested, by combining the changes and applying them to the initial version. We expect that our method can be easily added to the existing BPM system and, thereby, can improve the convenience of process modeling in an environment where a huge number of process models should be dealt with.

Acknowledgements. This work was supported by "Research Center for Logistics Information Technology (LIT)" hosted by the Ministry of Education & Human Resources Development in Korea.

References

1. Bae, H.: Develpment Integrated Logistics Process Management System (ILPMS) based on XML. PNU-Technical Paper-IS-2005-03, Pusan National University. (2005)
2. Conradi, R., Westfechtel, B.: Version Models for Software Configuration Management. ACM Computing Survey 30(2), 232–282 (1998)
3. Dittrich, K.R., Lori, R.: A. Version Support for Engineering Database Systems. IEEE Transaction on Softwate Engineering 14(4), 429–437 (1988)
4. Hunt, J. J., Vo, K. P., Ticky, W. F.: An Emperical Study of Delta Algorithms. In: Proceedings of ICSE'96 SCM-6 Workshop, LNCS, vol. 1167 (1996)
5. Katz, R.H.: Toward a Unified Framework for Version Modeling in Engineering Database. ACM Computing Surveys 22(4), 375–408 (1990)
6. Smith, H.: Business process management - the third wave: business process modeling language (bpml) and its pi-calculus foundations. Information and Software Technology 45(15), 1065–1069 (2003)
7. van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: a research agenda. Computers in Industry 53(3), 231–244 (2004)
8. WfMC: Workflow Management Coalition the Workflow Reference Model. WfMC Standards, WfMC-TC00-1003 (1995), http://www.wfmc.org