Journal of the Korean Institute of Industrial Engineers
Vol. 32, No. 2, pp. 126-132, June 2006.

BPM에서 관리되는 업무 프로세스의 버전관리

조은미 · 배혜림†

부산대학교 산업공학과

Version Management of Business Processes Managed by BPM

Eunmi Cho · Hyerim Bae

Department of Industrial Engineering, Pusan National University

Recently, business environments have been changing quickly. To establish competitive advantage, most enterprises have been using information systems such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM) and Customer Relationship Management (CRM). Many consider Business Process Management (BPM) a new innovative solution for enterprise-wide processes. As the BPM system is used more widely and matures, new techniques and functions will be developed by commercial vendors. However, they mainly focus on correctly executing process models, and user convenience has not been considered. In this paper, we have developed a new method of designing business processes, which provides users with an easy modeling interface. The method is based on version management. Version management of a process enables a history of the process model to be recorded. In order to prevent wasted storage, not all of the process versions are stored. An initial version and changes to each process are stored by automatically detecting changes. Our method enhances the convenience of the modeling business processes and thus helps the process designer. A prototype system is presented to verify the effectiveness of our method.

***Keywords*:** Business Process Management (BPM), Version Management, Process Model

1. Introduction

비즈니스 프로세스 관리(Business Process Management)는 조직 내/외부 전체 프로세스의 유기적 통합을 지원하고 프로세스의 계속적인 향상을 지원해 주는 경영혁신 방법론이다(Burlton, 2001). 이러한 방법론을 실현시키는 도구로서 조직의 프로세스를 정의하고 자동으로 실행하며 프로세스 모니터링 및 분석과 같은 기능을 실현하는 시스템을 비즈니스 프로세스 관리시스템(BPMS : Business Process Management System)이라 한다(Smith and Fingar, 2003). 최근, 비즈니스 프로세스 관리시스템을 도입하는 기업들이 증가하면서 기업 내부의 프로세스 관리뿐만 아니라, 기업간 프로세스를 모델링하고 관리하는 데에도 높은 관심을 갖고 있다. 이러한 환경에서 모델링 되는 프로세스들을 저장하고 추출하는 과정을 일괄적으로 관리하는 방법론으로 본 논문에서는 비즈니스 프로세스의 버전 관리 기능을 제안한다.

BPM의 적용영역이 점차 넓어지면서, BPM이 관리하는 비즈니스 프로세스모델에 대한 설계의 중요성이 높아지고 있다. 그러나, 기존의 프로세스 모델은 설계되는 과정에서 결과물을 단순 저장 및 추출하였다. 이러한 설계 방법론 하에서는 사용자, 특히 초보자가 프로세스 설계를 완벽하게 수행하기에는 어려움이 있어왔고, 프로세스 설계의 어려움은 BPM 도입의 장애 요인으로 인식되었다. 본 논문에서 제안하는 비즈니스 프로세스의 버전 관리 기능은 시간 흐름에 따른 프로세스 모델의 변경 및 저장을 체계적으로 관리할 수 있게 하고, 이를 통해 사용자의 편의를 도모하며, 부차적으로 프로세스 모델의 저장 공간을 최소화 할 수 있는 장점이 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문의 이론적 배경이 되는 비즈니스 프로세스 관리시스템, 버전관리 기법에 대해서 간략하게 살펴본다. 이어지는, 3장에서는 버전관

이 논문은 2005년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2005-003-D00468).
† 연락저자 : 배혜림 조교수, 609-735 부산시 금정구 장전동 산 30번지 부산대학교 산업공학과, Tel : 051-510-2733, E-mail : hrbae@pusan.ac.kr
2005년 12월 접수; 2006년 2월 수정본 접수; 2006년 2월 게재 확정.

리에 필요한 비즈니스 프로세스 모델을 제시하고, 4장에서는 비즈니스 프로세스 모델의 버전관리 기법을 제시하며, 버전관리 알고리즘을 제시한다. 5장에서는 버전관리 프레임워크를 구현한 프로토타입에 대해서 설명한다. 끝으로 6장에서 본 논문의 결론을 맺는다.

2. Backgrounds

2.1 Business Process Management Systems

기존의 워크플로우 관리시스템(WFMS : Workflow Management System)은 비즈니스 프로세스를 처리하기 위한 시스템 자동화와 관리에 주안점을 두고 있으며 BI(Business Intelligence)적인 요소는 최근까지 크게 고려하지 않아왔다(Bae *et al.*, 2004) . 하지만 근래, 워크플로우 벤더들은 기존 워크플로우 관리시스템(WFMS : Workflow Management System)이 간과하고 있는 프로세스 진단, 시뮬레이션 등의 기능을 추가하여 비즈니스 프로세스 관리시스템(BPMS : Business Process Management System)으로 그 영역을 확장 하고 있다(van der Aalst and Weijters, 2004). 가트너 그룹의 예상에 따르면 향후 BPM 의 확대와 함께 BPA(Business Process Analysis) 시장의 확장을 언급하고 있다(Gartner, 2002). 최근, BPA 툴로서 BAM (Business Activity Monitoring)이 주목을 받고 있으며, BAM은 시간적인 흐름에 따른 현 시스템의 운영현황을 시계열적으로 진단하고 관리하는데 주 목적이 있다(van der Aalst and Weijters, 2004).

기본적으로 업무프로세스의 표준화, 자동화 및 업무프로세스를 통제하고 관리한다는 점에서 BPMS와 WFMS는 크게 다르지 않다. BPMS의 주요요소를 Ovum(Ovum, 2002)에서 소개한 바에 따라 구성해보면 아래 <Figure 1>과 같다.

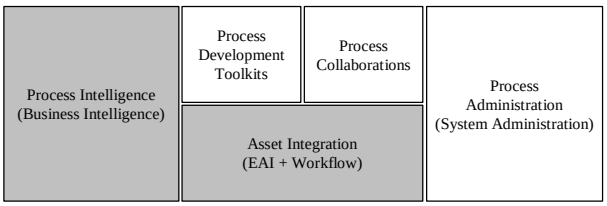

Figure 1. The compositions of BPMS.

<Figure 1>에서처럼 BPMS는 기존의 WFMS의 기능인 프로세스 표준화(Process Development Toolkits), 프로세스 통제(Process Collaborations) 및 관리(Process Administration)외에 WFMS가 간과 하고 있는 BI 측면이나 EAI(Enterprise Application Integration) 요소를 가미하여 그 영역을 확대하고 있으나, 이러한 노력들은 시스템 실행 단계에 국한되고 있으며, 모델링 단계 즉, 프로세스 모델의 디자인 단계에서의 시계열적인 관리 기법 또한 요구된다.

2.2 Version Management

버전 관리는 넓은 관점에서 시간의 흐름에 따라 변화하는 객체의 변화 양상을 체계적으로 관리하기 위한 방법론이다 (Katz, 1990). 버전관리에서 하나의 버전은 특정 시점에서의 관리 대상 객체의 상태에 대한 스넵샷(snap-shot)을 말한다.

버전 관리 기법에서 사용하는 모델은 대부분 객체의 변경 이력을 그래프로 표현하며 두 버전간의 관계는 리비젼(revision)과 배리언트(variant)로 표현할 수 있다(Conradi and Westfechtel, 1998). <Figure 2>에서 간단한 버전 그래프 예제로 두 개념을 살펴보면 다음과 같다.

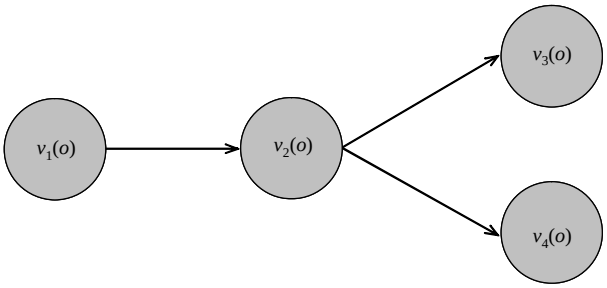

Figure 2. Version Graph.

- 리비젼 : 리비젼은 바로 이전 버전의 프로세스를 수정하여 새로이 생성된 프로세스 버전으로 정의할 수 있다. <Figure 2>에서 $v_2(o)$는 $v_1(o)$의 리비젼이다.
- 배리언트 : 이전 버전의 프로세스를 수정하여 두 가지의 새로운 프로세스가 생성된 경우 이후에 생성된 두 가지 프로세스의 관계를 배리언트라고 정의할 수 있다. <Figure 2>에서 $v_3(o)$는 $v_4(o)$의 배리언트이다.

버전관리 기법은 시계열 데이터베이스(temporal database) (Rodríguez *et al.*, 1999), 소프트웨어 소스코드 관리(software configuration management)(Conradi and Westfechtel, 1998), 설계 데이터 관리(engineering data management)(Dittrich and Lori, 1988) 등의 연구 분야에서 성공적으로 적용되었다. 적용 분야가 다양하지만 기본적으로 시간의 흐름에 따른 객체의 변화양상을 체계적으로 관리한다는 점에서는 모두 동일하다.

3. Business Process Model

본 장에서는 먼저 변경관리의 대상이 되는 프로세스 모델을 정의한다. BPM에서 사용되는 프로세스 모델은 프로세스(Process), 단위업무(Task), 링크(Link)등의 기본적인 객체와 그들의 속성으로 정의되는 것이 일반적이다. 본 연구에서 다루는 비즈니스 프로세스 모델은 변경관리의 대상이 되는 객체 및 속성만을 정의한다. 본 논문의 대상이 되는 프로세스 모델은 다음 정의1과 같다.

정의 1. Business Process Model

프로세스 집합 P의 원소인 하나의 프로세스 모델 p는 단위 업무 집합 T, 링크 집합 L, 속성집합 A로 구성되는 유방향 그래프 $p = (T, L, A)$로 정의 한다.

- 단위 업무 집합 $T = \{t_i \mid i = 1, \cdots, I\}$: t_i는 i-번째 단위업무, I은 단위업무의 총 개수이다.
- 링크 집합 $L = \{l_k = (t_i, t_j) \mid t_i, t_j \in T\ i \neq j\}$: l_k 는 t_i와 t_j 를 연결해 주는 링크이다.
- 속성 집합 A : 단위업무 혹은 링크의 속성들의 집합이다.

1) 단위업무속성 : $A_i = \{t_i.a_s \mid s = 1, \cdots, S_i\}$: S_i은 t_i가 가지는 속성의 총 수이다.
2) 링크속성 : $A_k = \{l_k.a_s \mid s = 1, \cdots, S_k\}$: S_k은 l_k가 가지는 속성의 총 수이다.

정의 1에 제시된 프로세스 정의는 본 논문의 과정을 논리적으로 설명하기 위해 단순화된 모델로 실제 프로세스는 상당히 복잡하며, 특히 분기와 병합 등의 요소들은 프로세스의 복잡성을 증대시킨다. 본 논문에서 고려하는 버전관리 방법론은 프로세스 모델이 복잡하여도 XML 기반의 마크업 언어로 표현이 가능하고 태그로 구분되는 객체를 인식할 수 있는 경우는 적용이 가능하다. 이에 본 논문은 표준화기구인 WfMC에서 제시하는 수준의 분기와 병합 구조를 가지는 프로세스 모델을 대상으로 본 연구의 접근법을 기술하기로 한다.

<Figure 3>은 인터넷 쇼핑의 주문 및 배송 프로세스 예제이다. 프로세스는 8개의 단위업무로 구성되어 있고 각 단위업무 및 링크는 속성을 가지고 있다. 먼저 주문 접수를($t_{주문접수}$) 받은 뒤 물품비용의 입금이 완료($t_{입금확인}$)되었는지 주문취소($t_{주문취소}$) 되는지를 확인하여 주문처리 여부를 결정한다. 주문이 취소 된 경우 agent에 의해 자동으로 취소 처리 후 담당자는 주문이 취소되었음을 고객에게 알린다($t_{주문취소\_알림}$). 입금이 확인된 경우, 상품을 준비($t_{상품준비}$)하고 배송사를 선택($t_{운송사연락}$)하여 연락한다. 모든 배송 관련 준비가 끝나면 상품을 발송($t_{상품발송}$)하고 고객이 물품을 받았는지 확인($t_{고객확인}$)되면 프로세스는 종료된다. 주문취소($t_{주문취소}$) 단위업무의 속성은 상품번호와 주문번호 ($t_{주문취소}.a_{상품번호}$, $t_{주문취소}.a_{주문번호}$) 라는 속성을 갖고 있고 l_8 의 링크 속성은 조건 제약인 주문취소요청($l_8.a_{조건주문취소요청}$)이라는 속성을 갖는다. 이러한 프로세스의 정의 시 모델을 제시하면 다음과 같다.

- 단위업무 집합 $T = \{t_{주문접수}, t_{입금확인}, t_{상품준비}, t_{주문취소}, t_{운송사연락}, t_{상품발송}, t_{주문취소알림}, t_{고객확인}\}$
- 링크 집합 $L = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_{10}, l_{11}\}$
- 속성 집합 $A = \{t_{주문접수}.a_{담당자}, t_{주문접수}.a_{접수문서}, t_{입금확인}.a_{고객번호}, t_{입금확인}.a_{계좌번호}, t_{입금확인}.a_{주문번호}, t_{상품준비}.a_{담당자}, t_{상품준비}.a_{고객번호}, t_{상품준비}.a_{상품번호}, t_{상품준비}.a_{고객주소}, t_{운송사연락}.a_{담당자}, t_{운송사연락}.a_{운송사\_전화번호}, t_{상품발송}.a_{담당자}, t_{상품발송}.a_{상품번호}, t_{주문취소}.a_{상품번호}, t_{주문취소}.a_{주문번호}, t_{주문취소알림}.a_{담당자}, t_{주문취소알림}.a_{고객\_전화번호}, t_{고객확인}.a_{담당자}, l_2.a_{조건\_입금자\_명단확보}, l_7.a_{조건\_발송완료\_확인}, l_8.a_{조건\_주문취소\_요청}, l_8.a_{조건\_주문취소\_처리확인}\}$

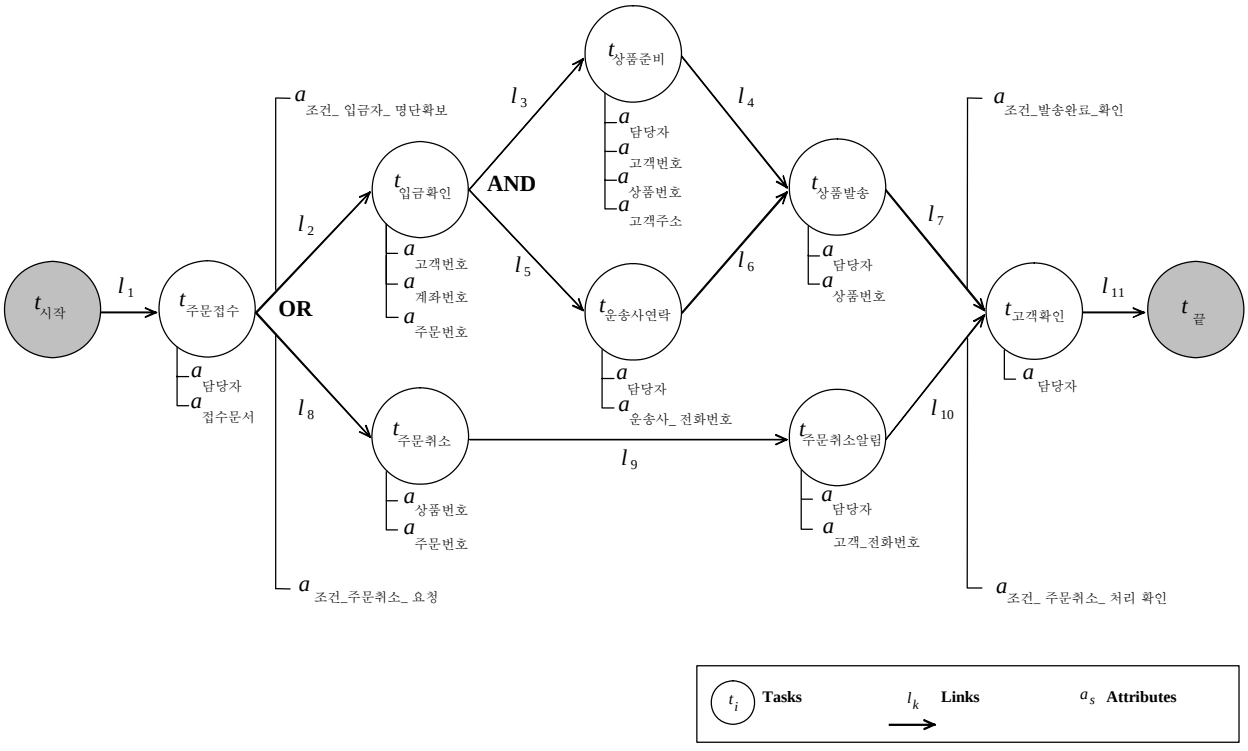

Figure 3. Internet shopping process.

4. Version Management of Business Process

본 장에서는 구체적인 프로세스 버전 관리 방법론을 설명한다. 이를 위하여 버전 관리의 대상이 되는 객체를 정의하고, 대상 객체의 변경 유형에 대하여 정의한다. 또한 체크인, 체크아웃 기법을 이용한 버전 관리 알고리즘을 제시한다.

4.1 Version Graph

BPM 시스템에서 프로세스를 설계하는 과정은 앞서 정의한 프로세스 모델을 완성해가는 과정이다. 이러한 과정에서 사용자는 단 한번에 완벽한 모델을 만들 수 없으며, 이전의 모델을 이용하여 좀 더 나은 모델을 만들어 간다. 모델의 변경은 모델을 이루는 구성요소에 대한 변경의 집합이며, 변경의 대상이 되는 하나의 객체 $o(\in O)$는 프로세스 모델을 이루는 구성요소들 중 하나이다. 즉, $o \in T$ 또는, $o \in L$ 또는, $o \in A$이다.

사용자가 프로세스 모델을 설계하는 과정에서 발생하는 새로운 모델의 결과물을 프로세스 버전이라고 하며, 버전 그래프는 프로세스를 대상 객체로 하며 프로세스 p의 버전을 $v(p)$로 표시한다. 이 때, 새로운 버전은 이전 버전에 대한 수정의 결과로 생성된다. 하나의 프로세스에서 대상 객체 단위마다 변경을 생성하며 객체 변경은 δ로 표현하고, 그 변경들의 집합을 프로세스 변경 Δ로 표현한다. 이 때, Δ는 각 객체들의 변경인 $\delta_q(o)$의 집합으로 표현되며 (즉, $\Delta = \{ \delta_q(o) \mid q=1,\cdots, Q \text{ and } o \in T, L, A\}$), 변경의 적용은 변경 적용 함수인 '$\cdot$'를 사용하여 표현한다. 만일, 프로세스 p의 m-번째 버전을 수정하여 n-번째 버전이 생성되었다면 두 버전 사이의 관계는 $v_n(p) = v_m(p) \cdot \Delta_{mn}$와 같이 표현된다. 그리고, 하나의 프로세스에 대한 변경의 생성과정은 다음에서 정의한 버전그래프로 표현한다.

정의 2. 버전 그래프(Version Graph, VG)

버전 그래프는 프로세스를 대상 객체로 한다. 프로세스 p에 대한 m-번째 버전을 $v_m(p)$라 할 때, 버전 그래프 VG는 하나의 프로세스 p에 대하여 $VG = (V, E)$와 같이 정의 된다. V와 E는 각각 노드와 아크의 집합을 의미한다.

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

위의 버전그래프에서 하나의 버전 $v_n(p)$이 버전 $v_m(p)$을 반복적으로 변경하여 생성가능할 때($\{(v_m(p), v_k(p)), (v_k(p), v_{k+1}(p)), \cdots, ((v_{k+1}(p), v_n(p))\} \subset E$), $v_n(p)$은 $v_m(p)$로부터 '도달가능(reachable)하다'라고 한다.

4.2 Process Model Modifications

프로세스 모델을 설계하는 사용자가 작업한 과정의 결과는 변경이며, 일반적으로 변경은 크게 세 가지의 범주로 정의된다. 즉, 사용자는 새로운 객체를 추가(ADD 연산)하거나, 기존 객체를 수정(MOD 연산)하거나, 삭제(DEL 연산)한다. 예를 들면, <Figure 2>에서 대상 객체 o가 단위업무 $t_{\text{주문접수}}$에 새로운 속성 $a_{\text{기한}}$으로 어떤 값이 추가되고 그 속성값이 추후에 변경될 때, 각 변경은 다음과 같이 표현한다.

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

프로세스를 설계하는 과정에서 변경 연산은 반복적으로 적용이 되지만, 반복적으로 적용된 변경 연산들은 하나의 변경 연산으로 표현할 수 있으며 이 과정은 다음과 같이 변경결합 연산'$\circ$'를 사용하여 나타낸다.

- $\delta_{new} = \delta_1(o) \circ \delta_2(o) = ADD\ t_{\text{주문접수}}.a_{\text{기한}}$("2006-01-24")

하나의 객체에 대한 변경의 반복 적용을 단일 변경으로 표현하기 위한 일반화된 규칙을 <Table 1>에 정리하였다. 변경 δ의 역변경은 δ^{-1}로 표현하며 빈변경은 $\delta_\emptyset$로 표현한다.

또한, 위에서 정의한 단위 변경간의 결합연산에는 다음과 같은 법칙들이 적용된다.

- 연산의 교환법칙은 성립하지 않는다.
  ($\delta' \circ \delta'' \neq \delta'' \circ \delta'$)
- 연산의 결합법칙은 성립한다.
  ($\delta' \circ \delta'' = \delta'' \circ \delta'$)
- 연산의 결합법칙은 성립한다.
  ($\delta' \circ (\delta'' \circ \delta''') = (\delta' \circ \delta'') \circ \delta'''$)
- 두 변경의 결합연산의 역연산은 각 변경의 역연산을 반대 순서로 적용한 것과 같다.
  ($(\delta' \circ \delta'')^{-1} = \delta''^{-1} \circ \delta'^{-1}$)

Table 1. Combination of changes

역 변 경		변경 결합	
추가연산의 역변경	$\delta_{ADD}^{-1} = \delta_{DEL}$	추가, 삭제연산의 결합	$\delta_{ADD} \circ \delta_{DEL} = \delta_\emptyset$
		추가, 수정연산의 결합	$\delta_{ADD} \circ \delta_{MOD} = \delta_{ADD}'$
삭제연산의 역변경	$\delta_{DEL}^{-1} = \delta_{ADD}$	삭제, 추가연산의 결합	$\delta_{DEL} \circ \delta_{ADD} = \delta_{MOD}$
수정연산의 역변경	$\delta_{MOD}^{-1} = \delta_{MOD}'$	수정, 수정연산의 결합	$\delta_{MOD} \circ \delta_{MOD}' = \delta_{MOD}''$
		수정, 삭제연산의 결합	$\delta_{MOD} \circ \delta_{DEL} = \delta_{DEL}$

객체 변경간의 결합연산을 바탕으로 변경집합간의 결합연산을 정의할 수 있으며 두 변경집합 Δ_1, Δ_2 간의 연산 ($\Delta_1 \circ \Delta_2$)은 각 집합에 포함된 객체변경들로 집합을 구성하되, 동일한 객체를 대상으로 하는 객체변경은 위의 결합연산에 의해 하나의 객체변경으로 표현한 집합이 된다.

4.3 Version Management Procedure

버전 관리는 정의 시 모델링 된 프로세스가 저장 되는 시점에 체크인을 통해 새로운 버전으로 저장된다. 새로운 버전을 생성하기 위해서 모든 버전을 저장하는 것은 저장 공간의 낭비를 가져오므로 이를 방지하기 위해 버전 관리 기법에 사용되는 델타기법(Hunt *et al.*, 1996)을 수정 적용하고 버전 생성을 체크아웃, 체크인의 두 과정을 통해서 구현한다. 델타기법은 전술한 변경 결합연산을 토대로 구현되며, 최초의 버전과 변경만을 저장하고 이를 토대로 원하는 버전의 프로세스를 구성할 수 있다.

프로세스의 버전 관리 방법론은 원하는 프로세스 모델을 공용 저장소에서 개인의 작업 영역으로 이동하는 체크아웃(Check-out)과, 개인 작업영역으로 이동해왔던 프로세스 모델을 다시 공용 저장소로 이동시키는 과정인 체크인(Check-in)을 통해서 이루어진다. 이 두 과정을 <Figure 4>에 순서도로 표현하였다.

체크아웃 프로시저는 특정한 프로세스 $v_m(p)$에 대한 체크아웃을 요구하면 최초 버전 $v_0(p)$로부터 해당 버전을 구성하는데 필요한 변경 집합들을 찾아서 결합 연산을 수행한다. 변경 집합간의 연산이 끝나면 최초 저장된 프로세스의 버전인 $v_0(p)$와 해당 버전을 구성하는데 필요한 변경들을 결합된 하나의 연산을 적용하여 원하는 프로세스 버전인 $v_m(p)$를 사용자에게 제공한다.

체크인 프로시저는 사용자가 프로세스를 수정하여 체크인을 요구하면 수정된 객체(단위업무, 링크, 속성)를 찾아서 어떠한 변경이 발생했는지를 삽입, 삭제, 수정 등의 내용으로 식별한다. 이들의 집합을 변경집합으로 기록하여 저장하고, 버전 그래프를 업데이트 한다.

4.4 Version Management Example

프로세스 버전이 실제 생성되는 예를 보이기 위해3장에서 예시한 <Figure 3> '인터넷 쇼핑 프로세스 모델'을 들어 설명한다. <Figure 3> 예제모델을 프로세스의 최초 버전인 $v_0(p)$라 하자. 이에 대하여 다음과 같은 변경이 발생할 수 있다.

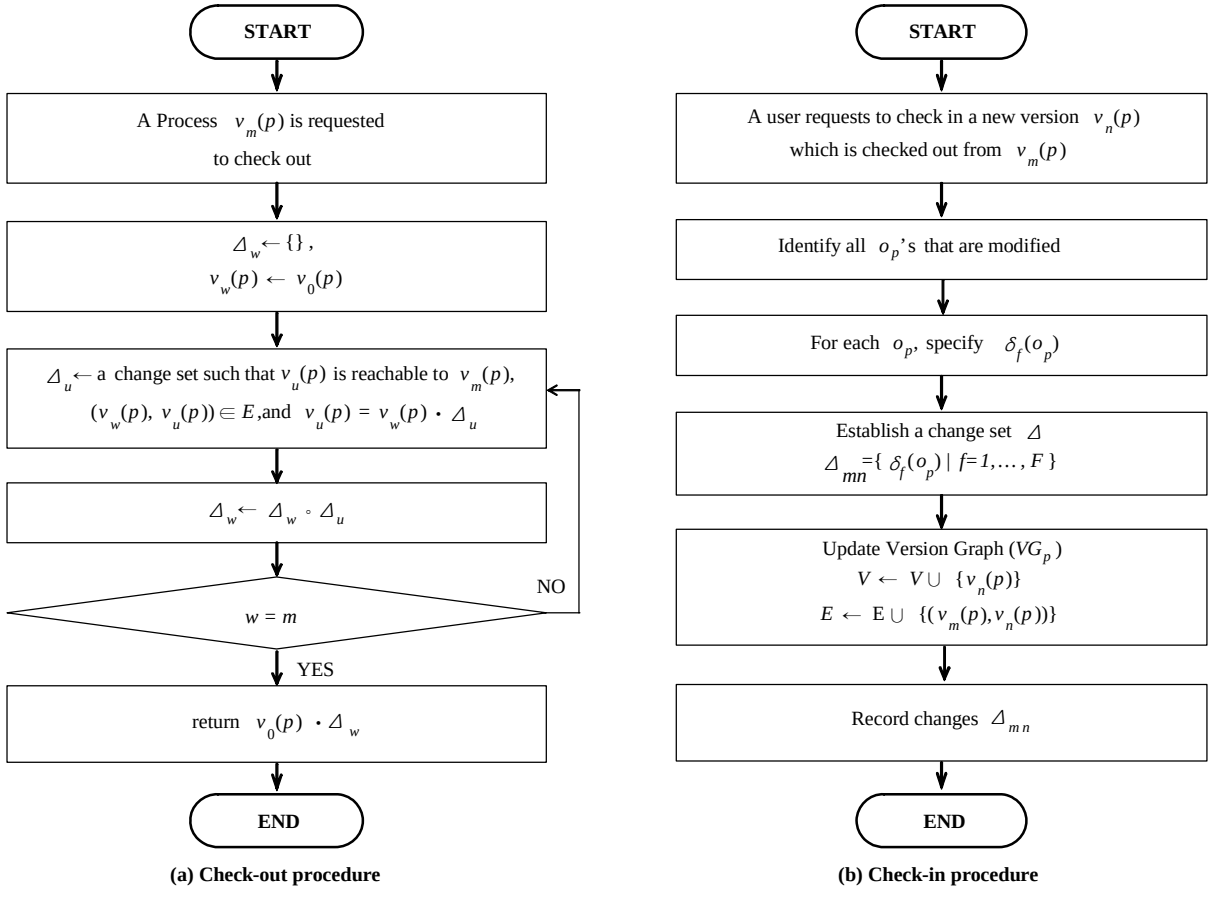

Figure 4. Flowcharts of check-out/in procedures.

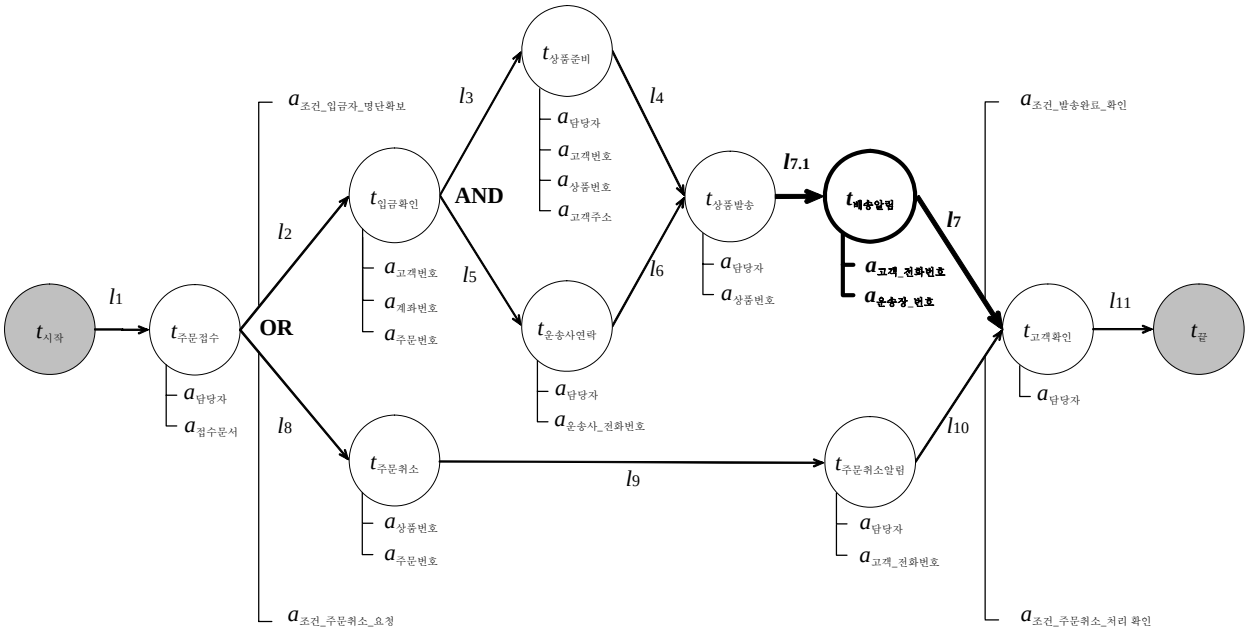

Figure 5. Modified process $v_1(p)$.

- 변경 1 : 프로세스 $v_0(p)$를 불러들여 단위업무 $t_{상품발송}$과 $t_{고객확인}$ 사이에 새로운 단위업무 $t_{배송알림}$를 추가한다. $t_{배송알림}$은 고객 전화번호와 운송장 번호($t_{배송알림}.a_{고객\_전화번호}$, $t_{배송알림}.a_{운송장\_번호}$)를 속성으로 갖는다. 링크 l_7 은 변경($t_{배송알림}$, $t_{고객확인}$) 되고 속성을 갖고 있지 않는 새로운 링크 $l_{7.1}$($t_{상품발송}$, $t_{배송알림}$) 가 생성된다.
  - $\delta_1(o_1)$ = *ADD* $t_{배송알림}$
  - $\delta_2(o_2)$ = *MOD* l_7(“ $t_{상품발송}$, $t_{고객확인}$”, “ $t_{배송알림}$, $t_{고객확인}$”), *ADD* $l_{7.1}$
  - $\delta_3(o_3)$ = *ADD* $t_{배송알림}.a_{고객\_전화번호}$(“82+051-633-0633”), *ADD* $t_{배송알림}.a_{운송장\_번호}$(“031-AG-08431-01”)
  - Δ_1 ={ $\delta_1(o_1)$, $\delta_2(o_2)$, $\delta_3(o_3)$}
- 변경 2 : 변경 1에서 생성된 $v_1(p)$를 불러들여 $t_{상품준비}$ 단위업무의 담당자 속성($a_{담당자}$)을 “김삼순”에서 “이희진”으로 변경한다.
  - $\delta_4(o_4)$ = *MOD* $t_{상품준비}.a_{담당자}$(“김삼순”, “이희진”)
  - Δ_2 ={ $\delta_4(o_4)$}
- 변경 3 : 1단계 수정에서 생성된 $v_1(p)$를 불러들여 $t_{운송사연락}$ 단위업무의 속성($a_{운송사\_전화번호}$)을 “82+051-123-4567”에서 “82+051-987-6543”으로 변경한다.
  - $\delta_5(o_5)$ = *MOD* $t_{운송사연락}.a_{운송사\_전화번호}$(“82+051-123-4567”, “82+051-987-6543”)
  - Δ_3 = { $\delta_5(o_5)$}

최초버전 $v_0(p)$에서 Δ_1변경을 통하여 $v_1(p)$가 생성이 되며, $v_1(p)$에 대하여 각각 Δ_2, Δ_3을 통하여 $v_2(p)$, $v_3(p)$ 가 각각 생성된다. 이 때, $v_2(p)$는 $v_1(p)$의 리비젼이고 $v_3(p)$는 $v_2(p)$의 배리언트가 된다. 변경이 발생한 후 체크인하는 과정에서는 각 변경이 탐지되어 변경만을 저장하고<Figure 6>과 같은 버전그래프를 업데이트한다. 체크아웃 과정에서는 저장된 변경과 최초버전을 통해 원하는 버전을 구성하게 된다. 예를 들면 $v_2(p)$ = $v_0(p)$ • ($\Delta_1 \circ \Delta_2$)를 통하여 얻어진다.

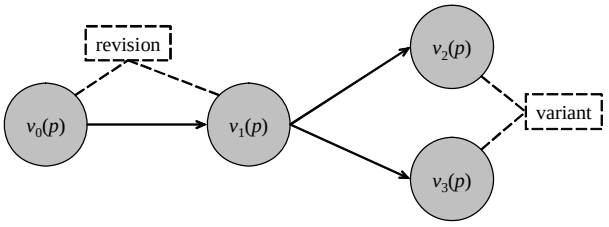

Figure 6. Version Graph of ‘internet shopping’.

5. Prototype System

프로세스 정의 시 생성되는 프로세스 모델의 변경을 체계적으로 관리하기 위한 프로세스 버전 관리 시스템의 프로토타입을 개발하였다. 제안된 방법론은 BPM시스템인 ILPMS(Integrated Logistics Process Management System)(Bae, 2005)의 정의 시 모듈인 프로세스 디자이너에 구현하였다. 전체 시스템의 구조는 <Figure 7>과 같다. 그림에서 보는 바와 같이 사용자는 프로세스 정의도구를 활용하여 프로세스 모델을 설계하고 버전관리모듈과의 인터페이스를 통해 체크아웃 및 체크인을 수행한다. 설계된 프로세스 모델은 프로세스DB에 최초버전과 변경으로 저장이 되고 원하는 시점에 해당버전으로 구성되어 사용자에게 전달된다.

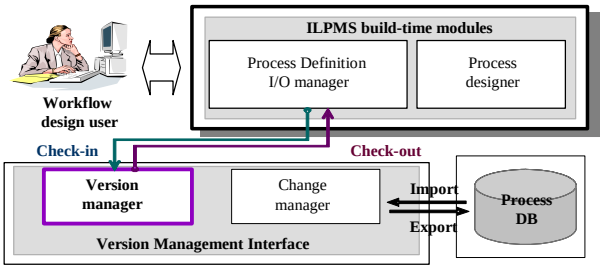

Figure 7. System architecture.

사용자들은 모델러 모듈에서 제공하는 인터페이스를 이용하여 편리하게 프로세스를 수정 및 변경할 수 있으며, 체크인을 통해 프로세스의 버전을 자동으로 생성한다. 이러한 과정을 표현하는 실제 화면의 예를 <Figure 8>과 <Figure 9>에 나타냈다.

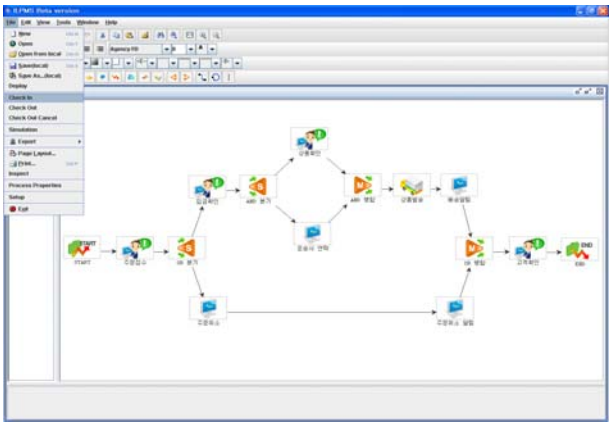

Figure 8. The screen of check out.

<Figure 8>은 사용자가 체크아웃 버튼을 클릭하여 원하는 프로세스를 불러온 뒤 프로세스에 수정을 가한 후 체크인을 선택하여 프로세스를 저장하는 화면이다. 버전 관리 시스템은 변경된 프로세스를 새로운 버전으로 저장하고, 버전그래프를 업데이트 한다. 새로운 버전이 업데이트 되는 화면을 <Figure 9>에 나타내었다.

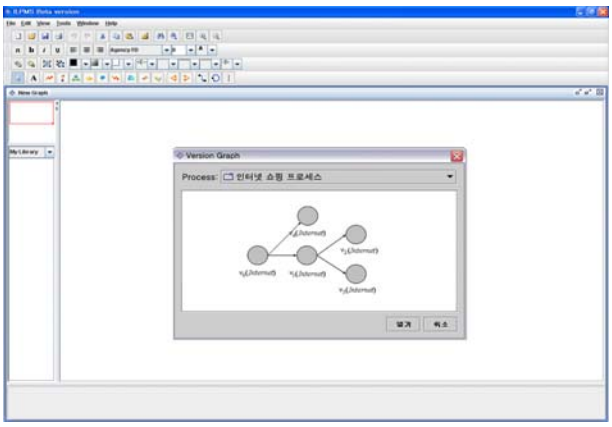

Figure 9. The screen of version graph.

6. Conclusions

본 연구에서는 BPM에서 사용자가 편리하게 프로세스를 모델링 할 수 있는 방법으로 버전관리 기법을 제안하였다. BPM시스템 적용의 장애요소로 프로세스 설계의 어려움이 지적되고 있으며, 초보자와 일반 사용자도 프로세스를 설계할 수 있도록 하는 것은 매우 중요한 문제이다. 버전 관리방법론을 적용하면, 기존 버전이 사용자에게는 프로세스 템플릿과 같은 역할을 함으로써 프로세스 설계를 편리하도록 해준다. 따라서, 기존에 설계된 프로세스를 통하여 편리하게 프로세스를 설계할 수 있도록 한다는 점에서 본 연구의 의의를 찾을 수 있다. 본 논문에서 제시한 프로세스 버전관리 방법론은 각 프로세스에 대하여 단위업무, 링크, 및 이들의 속성이 어떻게 변했는지를 기록함으로써 변경을 체계적으로 관리하고 원하는 시점의 프로세스 버전을 쉽게 구성할 수 있다. 또한 변경 부분만을 저장하므로 인해 일반적인 버전관리방법론이 가지는 저장공간의 낭비 문제를 해결하였다.

References

Bae, H. (2005), Develpment Integrated Logistics Process Management System (ILPMS) based on XML, *PNU-Technical Paper-IS-2005-03*, Pusan National University.

Bae, H., Hur, W., Yoo, W., Kwak, B., Kim, Y., and Park, Y. (2004), Document Configuration Control Processes Captured in a Workflow, *Computers in Industry*, **53**(2), 117-131.

Burlton R. T. (2001), *BPM-Profiting process management*, SAMS, USA.

Conradi, R. and Westfechtel, B. (1998), Version Models for Software Configuration Management, *ACM Computing Survey*, **30**(2), 232-282.

Dittrich, K. R. and Lori, R. A. (1988), Version Support for Engineering Database Systems, *IEEE Transaction on Software Engineering*, **14**(4), 429-437.

Gartner (2002), Gartner's Application Development and Maintenance Research Note M-16-8153, The BPA Market Cashes another Major Updraft, http://www.gartner.com/

Hunt, J. J., Vo, K. P., and Ticky, W. F. (1996), An Empirical Study of Delta Algorithms, *Proceedings of ICSE'96 SCM-6 Workshop*, LNCS 1167.

Katz, R. H. (1990), Toward a Unified Framework for Version Modeling in Engineering Database, *ACM Computing Surveys*, **22**(4), 375-408.

Ovum (2002), Technical Note, BPM : a System Solution to Crisis, http://www.ovum.com

Rodríguez, L., Ogata, H., and Yano, Y. (1999), TVOO : A Temporal Versioned Object-Oriented data model, *Information Science*, **114**(1-4), 281-300.

Smith H. and Fingar P. (2003), Business Process Management : The Third Wave, *Journal of Information System*, **18**(1), 128-131.

van der Aalst, W. M. P. and Weijters, A. J. M. M. (2004), Process mining : a research agenda, *Computers in Industry*, **53**(3), 231-244.