e-Business를 위한 BPM 문서 변경관리 모델

A change Management Model for BPM Documents in e-Business Environments

배혜림(Hyerim Bae)**, 조재균(Cho, Jae Gyun)**, 정석찬(Jeong, Seok-chan)**, 박기남(Park, Ki-Nam)**

초 록

비즈니스 프로세스 자동화 (Business Process Management, BPM)는 이기종 시스템간의 통합을 추구하면서 비즈니스 프로세스의 수명주기를 체계적으로 관리하기 위한 최근의 정보시스템 추세중의 하나이다. 이러한 BPM은 e-Business와 같이 다수의 조직 및 기업이 상호작용 하여 이루어지는 복잡한 프로세스의 자동실행을 위해서는 반드시 필요한 요소로 인식되고 있다. 다조직 프로세스에서는 각 비즈니스 주체들간에 자원의 변경과정을 이해하고 명확히 하는 것이 중요하다. 그러나, 부족한 용량, 프로세스 구조의 복잡성 및 모델의 부재로 인하여 자원의 변경관리가 어려운 문제로 인식되어 왔다. 본 논문에서는 새로운 프레임워크를 제안하여 비즈니스 프로세스 자원의 가장 중요한 요소 중 하나인 문서의 변경을 지원한다. 이를 위하여 기존 프로세스 자동화 도구에서와 같이 프로세스 수명주기를 '정의시' 와 '실행시' 의 두가지로 구분하고 프로세스 진행과정에서 문서의 변경을 관리할 수 있는 새로운 모듈을 도입하여 추가한다. 제안된 방법론은 문서 구조모델, 프로세스 구조모델, 문서와 프로세스간 연계모델과 버전 저장모델을 제공한다. 이러한 프레임워크의 유용성을 보이기 위하여 원형을 구현하여 본 논문에 제시한다.

ABSTRACT

Business Process Management (BPM) is an emerging trend of managing business process life cycles and integrating heterogeneous systems. Recently, BPM is considered as an essential element for automation of complex business processes involving many companies, particularly for those in an e-Business environment. In such an environment, it is very important for each business partner to trace history of resources. However, it has been a difficult problem for Workflow Management Systems(WFMSs) to support management of resource changes because of limited storage, complex process structure, and absence of formal change model. In this paper, a new framework is proposed, which can support change management for documents, one of core resources of business process. Under a traditional WFMSs framework, all workflow components belong to either build-time or run-time function, and executions of workflow processes are performed by the two functions. To manage changes of documents while process executing, a new framework with additional component modules are required. A version control method is introduced for the purpose of managing document changes. The proposed method includes five models for document structure, process structure, association between process and document, version management, and efficient version storage. A prototype system is developed to demonstrate the effectiveness of the proposed models.

키워드 : 비즈니스 프로세스 관리, 워크플로우, XML, 버전관리

Business Process Management, Workflow, XML, Version Management

---

* 이 논문은 2003년도 동의대학교 우수연구자 지원사업 교내연구비를 지원받아 연구되었음.

** 동의대학교 인터넷비즈니스학과

1. 서 론

본 연구의 목적은 e-Business에서 사용되는 자원들의 변경을 체계적으로 관리하는 새로운 모델을 개발하는데 있다. 최근의 경영환경은 네트워크의 비약적인 발달과 비즈니스 모델의 다양화로 인해 점점 더 복잡해지고 있다. 이는 기업이 수행해야 하는 프로세스의 규모가 커지고 복잡해졌다는 것을 의미하며, 여러 기업이 협력하여 프로세스를 수행해야 함을 의미한다. 따라서, 여러 기업이 관여하는 복잡한 프로세스를 가지는 e-Business에서는 프로세스를 체계적으로 관리하고 그 수행을 지원하는 시스템적 접근법이 필요하다. BPM(Business Process management) 방법론이나 ebXML과 같은 표준 프레임워크는 이러한 문제를 시스템적으로 접근하는 최근의 조류라고 할 수 있다. 본 연구에서는 e-Business에서 발생하는 복잡한 프로세스에서 두 가지 중요한 측면을 고려하고, 이들을 정보시스템 기술을 통하여 해결하고자 한다. 먼저 프로세스 자체의 올바른 수행과 관리를 고려한다. 프로세스를 컴퓨터가 실행 가능한 형태로 표현한 후 이를 토대로 해당 프로세스의 조정, 제어, 관리를 담당함으로써 프로세스 수행을 자동화하는 것이 필요하다. 다음으로 고려할 사항은 프로세스를 통하여 다양한 형태로 발생하는 자원의 효과적인 관리이다. 프로세스의 수행을 위해서는 각종 데이터와 문서를 사용자들이 공유, 사용하게 되며 이에 대한 체계적인 관리가 필요하다. 특히, 대부분의 비즈니스 프로세스는 문서작업을 포함하고 있어서 이에 대한 변경관리는 필수적이다.

e-Business와 같이 규모가 크고 복잡한 프로세스가 수행되는 동안 문서는 생성이 되고, 수정과정을 반복적으로 거쳐 프로세스의 종료와 함께 완성된다. 따라서 프로세스의 진행과정은 곧 문서의 완성과정으로 볼 수 있으며, 문서자원의 변경과정과 프로세스의 진행을 연계하여 체계적으로 관리할 필요가 있다. 이를 위해서는 공유자원으로서의 문서를 프로세스 중심으로 바라보고 이에 대한 변경을 관리하여야 한다.

본 연구에서는 프로세스가 사용하는 자원 중 가장 일반적으로 사용되는 문서에 대하여 그 변경을 프로세스의 진행과 연계하여 관리할 수 있는 새로운 방법론을 제시한다. 이러한 방법론은 버전 관리 기법에 기반을 두고 있으며, 문서 및 프로세스의 구조, 문서와 프로세스간의 연계, 프로세스 수행에 따른 버전 자동생성, 버전의 저장 및 재구성, 그리고 버전에 대한 추출 기능을 포괄적으로 제공한다. 본 연구에서 제공하는 방법론의 핵심요소는 프로세스가 수행되는 동안 발생하는 버전 생성이 프로세스의 의미를 반영하여 자동으로 이루어지는 데 있으며, 생성된 버전 정보를 응용하면, 버전의 저장과 질의가 효과적으로 이루어질 수 있다.

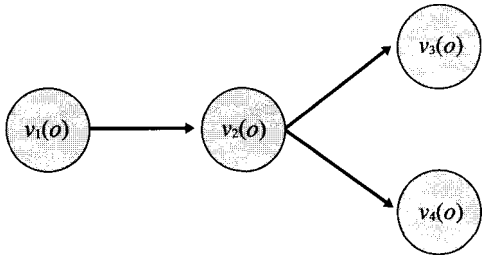
본 논문의 방법론은 여러 부서 혹은 회사가 함께 관여하여 프로세스를 진행하는 경우 효과적이다. 여러 부서가 관여하는 경우에는 다른 부서가 작업한 결과를 토대로 문서작업을 수행해야 하므로 각 부서가 작업한 내용의 이력을 기록하는 것이 문서의 효율적인 이용에 있어 중요하다. 즉, 프로세스가 종료된 후에도

전체 프로세스를 통해 발생한 문서의 변경이력을 추적할 수 있을 뿐 아니라, 프로세스의 이상종료에 의한 회복 요구가 발생한 경우에도 이를 토대로 문서의 상태를 복구할 수 있는 기반을 마련하게 된다.

2. 이론적 배경

2.1 버전관리 기법

본 논문에서는 프로세스 자원의 변경을 다루기 위하여, 이미 이론적 기반을 확보하고 널리 적용되고 있는 버전관리 기법을 응용한다. 넓은 의미에서 버전관리 기법은 시간에 따른 객체의 변화를 다루는 방법론이라 할 수 있으며, 버전이란 특정 시점에서 의미를 가지는 객체의 상태라고 정의할 수 있다[9]. 버전관리 기법에서 사용하는 모델은 대부분 객체의 변경 이력을 그래프로 표현하는 경우가 많으며[4, 14] 다음 그림 1에 간단한 버전 그래프를 표현하였다. 그림에서 각 버전간의 관계는 두 가지로 표현되는데 하나는 리비젼(revision)이며 다른 하나는 배리언트(variant)이다. 먼저 리비젼관계는 이전 버전의 수정에 의해서 새로운 버전이 생성되는 경우, 이들 두 버전간의 관계를 표현한다. 예를 들면, $v_2(o)$는 $v_1(o)$의 리비젼이며 $v_3(o)$는 $v_1(o)$의 리비젼이다. 한편, 배리언트는 하나의 문서를 서로 다른 방식으로 수정하여 각기 다른 대안이 될 수 있는 두 버전간의 관계이다. 예를 들면 그림에서 $v_3(o)$는 $v_4(o)$의 배리언트이다.



〈그림 1〉 기본 버전 모델

버전 관리와 관련하여 많은 기존 연구들이 이루어져 왔으며, 활발하게 버전 기법이 적용된 분야는 SCM (Software Configuration Management) [4, 13], 설계 데이터 관리 [9, 10, 11], 그리고 시계열 데이터베이스[12] 등이다. SCM분야의 변경관리는 소프트웨어 개발 프로젝트를 진행함에 있어 소스코드들의 변경을 체계적으로 관리함으로써 개발의 효율을 높이려는 시도이다. 설계 데이터 관리는 CAD도면과 같은 공학 데이터의 변경과 컨피규레이션을 관리함으로써 다중 사용자 환경에서 효과적으로 제품을 개발하려는 시도이다. 마지막으로 시계열 데이터베이스는 데이터베이스에 저장되는 단순데이터에 대하여 시간을 축으로 그 변경을 기록함으로써, 사용자가 시간을 매개변수로 질의를 수행할 수 있도록 해준다. 그러나 이러한 세 가지 영역에서의 변경관리 연구는 단순히 시간의 흐름에 따라 자원의 변경을 기록하는 것으로 공식화된 프로세스와의 연관을 생각하지 않은 방법론이다.

이들 기존 연구 중에서, SCM에서의 버전관리는 반복적인 작업의 결과로 발생하는 객체의 변화를 관리한다는 측면에서 본 연구에서 고려하고 있는 기법과 유사하다. 그러나,

SCM의 버전 기법은 단순한 시간에 의한 객체 변경을 다룰 뿐 프로세스 관점에서 객체의 변경이력을 관리하지 못한다는 단점을 지니고 있다. SCM 관련 연구에서 이미 프로세스의 개념을 버전관리에 도입할 것을 언급한 사례가 있으나[4] 실제적인 방법론이 제시되지 못하고 있다. 이는 SCM에서 대상으로 삼고 있는 소프트웨어 개발 프로세스가 정형화된 프로세스 모델을 갖추고 있지 않은 데서 기인한 것이다. 이와는 달리, 워크플로우 관리시스템은 매우 정형화된 프로세스 모델을 갖추고 있으며, 이는 버전관리를 위한 훌륭한 기반을 제공한다고 할 수 있다.

2.2 프로세스 관리: 워크플로우

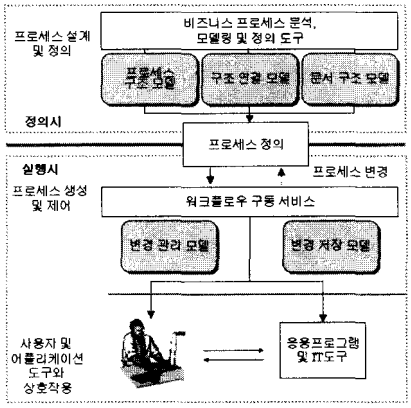
워크플로우 관리는 비즈니스 프로세스 자동화와 관련된 용어이며, 워크플로우 관리시스템 (Workflow Management System, WfMS)은 워크플로우 관리를 구현하는 소프트웨어 시스템으로 최근 그 적용영역을 넓혀가고 있다 [6]. 프로세스 자동화는 비즈니스 프로세스의 표현을 컴퓨터가 이해할 수 있는 형태로 정의한 프로세스 모델에 기반을 두고 있으며, 워크플로우 관리 시스템은 프로세스 모델을 이용하여 프로세스를 정의한 후 이를 실행하는 것을 통해 프로세스를 자동화하고 프로세스의 생명주기를 통제, 관리한다.

워크플로우 관리시스템이 관리하는 대부분의 비즈니스 프로세스는 문서작업을 포함하고 있으며, 상용 제품들은 이들 문서를 관리하기 위한 내장함수를 가지고 있다. 이러한 내장함수를 통해서 프로세스가 진행되는 동안 문서의 전달 및 저장과 같은 단순한 기능을 제공한다. 워크플로우 관리시스템에서 문서를 관리하는 다른 방법은 기존의 문서관리 시스템 (EDMS, Electronic Document Management System)과 인터페이스 하는 방식이다. 이러한 방식은 이미 문서관리 시스템을 구축하여 사용하고 있는 기업이 워크플로우를 도입하는 경우 유용하며 두 시스템이 상호 독립적일 수 있다는 점에서 효과적이다. 그러나, 이 두 가지 방식은 모두 문서의 최근 버전만을 유지한다는데 한계가 있으며 프로세스의 관점에서 문서의 변경을 관리하지 못한다.

3. 프로세스 수행과 문서 변경 관리

워크플로우 관리시스템은 정의시 기능과 실행시 기능으로 나누어 이해하는 것이 일반적이다. 워크플로우 관련 표준을 관장하는 WfMC (Workflow Management Coalition)에서도 워크플로우를 구성하는 각 표준 컴포넌트와 컴포넌트들 간의 인터페이스를 두 가지 기능의 틀 위에서 설명하고 있다[8].

먼저, 정의시 기능은 프로세스 모델을 정의하여 프로세스가 수행될 수 있는 준비를 하는 단계의 기능이다. 이를 위하여 프로세스를 다수의 단위업무들로 나누고 이들 간의 선후 관계 및 프로세스를 수행하는 데 필요한 속성을 정의한다. 이렇게 정의된 프로세스는 각 워크플로우 관리시스템이 사용하는 고유의 프로세스 모델로 저장이 된다. 다음으로 실행시 기능은 프로세스 모델을 기반으로 프로세스



〈그림 2〉 문서변경관리를 위한 워크플로우 모델 ([8]으로부터 수정)

를 시작시키고, 제어, 관리하는 데 관련된 기능이다. 실행시 기능의 핵심은 워크플로우 구동 서비스이며 특히, 시스템의 중앙에 위치한 워크플로우 엔진이라고 하는 서버모듈이 이를 담당한다.

본 연구에서는 프로세스를 관리하고 실행해주는 워크플로우 관리시스템이 문서의 변경을 체계적으로 관리하기 위한 새로운 모델들을 제안한다. 그림 2에서 보는 바와 같이 프로세스가 수행되는 동안 문서의 변경을 체계적으로 관리하기 위해서는 다섯 개의 세부 모델이 필요하다. 각 모델에 대하여 그 기능을 소개하면 다음과 같다.

- 프로세스 구조모델: 프로세스의 구조를 표현하는 모델로 하나의 프로세스를 단위업무로 쪼개고, 이들간의 상호 연결 관계를 네트워크로 표현하여 프로세스의 수행 순서를 정의한다.
- 문서 구조모델: 프로세스의 진행과 연계하여 문서 작업을 수행하는 데 필요한 문서의 구조를 정의한다. 즉, 문서를 작업의 단위가 될 수 있는 영역으로 분할한다.
- 구조 연결모델: 프로세스 구조모델과 문서 구조모델의 결과로 만들어진 프로세스와 문서가 상호 연관을 가지게 함으로써 실행시에 원하는 문서작업이 프로세스와 연계되어 이루어질 수 있도록 한다. 즉, 분할된 각 문서의 영역을 프로세스의 단위업무에 적절히 할당한다.
- 변경 관리모델: 정의시에 주어진 프로세스 구조모델, 문서 구조모델, 그리고 구조 연결모델을 바탕으로 실행시에 문서의 변경을 기록, 관리해주는 모델이다. 이는 프로세스의 구조 및 유형을 반영하여 버전이 자동으로 생성되도록 해준다.
- 변경 저장모델: 하나의 문서에 대하여 여러 버전이 생성됨으로 인해 발생하는 저장공간의 낭비를 최소화하는 모델이다. 본 연구에서는 변경 저장모델을 통하여 버전간의 변화분만을 저장하고 이를 통하여 문서가 효율적으로 저장되도록 한다.

위에서 기술한 세부 모델중 프로세스 구조모델, 문서 구조모델, 구조 연결모델은 정의시에 이루어지는 모델에 해당하며, 변경 관리모델, 변경 저장모델은 실행시 기능에 해당한다. 각 모델에 대해서 이어지는 두 장에서 상세히 설명한다.

4. 정의시 모델

4.1 프로세스 구조모델

대부분의 상용 워크플로우 관리시스템은

프로세스 모델을 보유하고 있으며, 일반적인 프로세스 모델은 전체 프로세스와 이를 구성하는 단위업무와 이들 간의 선후 관계, 그리고 이들 각각을 표현하는 다수의 속성들로 구성된다 [8]. 본 연구는 프로세스 모델 자체를 다루고 있지 않으므로 본 논문의 이해를 돕는데 요구되는 구조적 수준의 프로세스 모델만을 논하며, 보다 자세한 프로세스 모델에 대해서는 [2, 8]등을 참고하기 바란다.

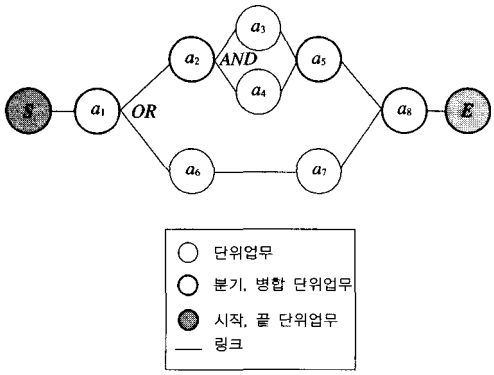
본 논문에서 사용하는 프로세스 구조모델은 다음과 같다.

정의 1. 프로세스 구조모델

프로세스는 다음과 같이, 유방향 그래프 $p=(A, L)$와 분기 및 병합을 위한 라벨링 함수 $f(o)$로 정의된다.

- 단위 업무 집합 $A = \{a_i \mid i = 1, 2, 3, \ldots, N\}$. 단, a_i는 i-번째 단위업무이고 N은 단위업무의 수이다.
- 링크 집합 $L = \{(a_i, a_j) \mid a_i, a_j \in A$, 그리고, $(i \neq j)\}$. 단, (a_i, a_j)는 단위업무 a_i는 단위업무 a_j보다 바로 앞서 실행되어야 함을 의미한다.
- 분기 단위업무 a_i($S = \{a_k \mid (a_i, a_k) \in L\}$, $|S| > 1$) 에 대하여, 만일 모든 a_k가 수행되어야 한다면 $f(a_i) =$ 'AND' 이고 그렇지 않으면 'OR' 이다.
- 병합 단위업무 a_j($P = \{a_k \mid (a_k, a_j) \in L\}$, $|P| > 1$) 에 대하여, 만일 모든 a_k가 수행되어야 한다면 $f(a_j) =$ 'AND' 이고 그렇지 않으면 'OR' 이다.

위의 정의를 이용한 간단한 프로세스 모델의 예를 그림 3에 제공하였다. 이 때, 단위업무 집합 $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$가 되고 링크집합 $L = \{(a_1, a_2), (a_2, a_3), (a_2, a_4), (a_3, a_5), (a_4, a_5), (a_1, a_6), (a_5, a_8), (a_6, a_7), (a_7, a_8)\}$이다. 또한, 두 개의 분기 단위업무 a_1, a_2와 두 개의 병합 단위업무 a_5, a_8를 가져서 $f(a_1) = f(a_8) =$ 'OR', $f(a_2) = f(a_5) =$ 'AND' 가 된다.



〈그림 3〉 프로세스 구조 모델

4.2 문서 구조모델

프로세스 구조모델이 프로세스의 구조를 표현하는 데 반해 문서 구조모델은 문서의 구조를 표현한다. 비즈니스 영역에서 사용되는 문서들은 대부분 정해진 양식을 바탕으로 잘 정형화되어 있는 것이 일반적이다. 본 연구에서는 이러한 비즈니스 문서의 특성을 반영하여 정형화된 문서의 일정 영역이 프로세스의 단위업무별로 작성될 수 있도록 문서를 구조화한다.

그림 4에 예로 든 비즈니스 문서는 프로세스의 단위업무에서 한꺼번에 처리 가능한 단

위로 구조화되어 있다. 본 논문에서는 이러한 문서의 논리적인 단위를 작업단위(work unit)라 부르며 워크플로우 프로세스에서 각 단위 업무가 한꺼번에 처리할 수 있는 문서작업의 단위이다. 그림의 예에서 특정 단위업무가 '작업단위-1'을 처리하고 나면 다음 단위업무가 이어서 '작업단위-2'를 처리하는 방식으로 문서가 처리된다. 결국 프로세스가 완료되는 시점에서는 문서도 완성된다고 볼 수 있다.

그림 4의 예에서 보듯이 문서는 계층적으로 구조화되며 이를 다음과 같이 표현한다.

정의 2. 문서 구조모델

하나의 문서 d는 작업단위들의 집합이며 각 작업 단위는 몇 개의 필드로 구성되어진다.

- $d = \{w_m \mid m = 1, \cdots, M\}$. 단, w_m은 문서 d의 m-번째 작업단위이고, M은 작업단위의 수이다.
- $w_m = \{f_l \mid l = 1, \cdots, L_m\}$. 단, f_l은 w_m의 l-번째 필드이고 L_m은 w_m에 있는 필드의 수이다.

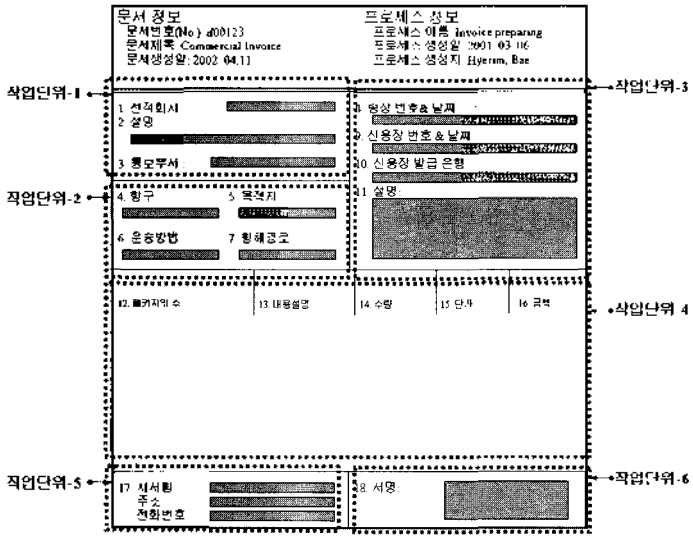
여기서, $w_m \cap w_n = \emptyset$ $(w_m \neq w_n)$,이다.

4.3 구조 연결모델

또 다른 정의시 모델로 구조 연결모델이 있는데 이는 실행시에 문서가 프로세스의 진행과정과 연계하여 자동으로 버전이 생성되도록 하기 위한 모델이다. 이를 위하여 [3]에서 제안한 문서-프로세스간 연계모델을 사용하기로 한다. 본 논문의 전개에 필요한 구조 연결모델은 다음과 같이 정의되며, 이에 대한 상세한 표현법과 자세한 사용법에 대해서는 [3]을 참고하기 바란다.

정의 3. 문서-프로세스 연관

문서-프로세스 연관은 두 개의 화살표 '↑'



〈그림 4〉 비즈니스 문서의 예

와 '↓'를 이용하여 표현하며, 다음과 같은 두 가지 유형의 연관이 있다.

- $\uparrow {}^{W}_{A}$는 특정 단위업무에 할당된 모든 작업단위를 표현한다.
- $\downarrow {}^{W}_{A}$는 특정 작업단위가 할당된 모든 단위업무를 표현한다.

예를 들어, 문서 d의 두 작업단위 w_1, w_2가 프로세스 p의 단위업무 a_1에 할당되고 w_1은 a_2에 한번 더 할당되었다면, $p.a_1 \uparrow {}^{W}_{A} * = \{d.w_1, d.w_2\}$이 되고, $d.w_1 \downarrow {}^{W}_{A} * = \{p.a_1, p.a_2\}$이 된다.

5. 실행시 모델

5.1 변경 관리모델

본 연구에서 실행시에 프로세스 단위업무에서 이루어지는 문서에 대한 작업의 결과는 버전으로 생성, 관리된다. 즉 워크플로우 관리 시스템이 문서 작업에 의한 변경을 자동으로 반영하고 이를 버전으로 생성한다. 이러한 버전생성은 프로세스의 구조와 패턴을 바탕으로 이루어지며 버전 그래프 및 각 버전 간의 연관으로 그 결과를 기록한다. 먼저 이를 위한 버전 그래프는 다음과 같이 정의된다.

정의 4. 버전 그래프

버전 그래프 VG는 하나의 객체 o에 대하여 $VG = (V,E)$와 같이 정의 되며, V와 E는 각각 노드와 아크의 집합을 의미한다.

- $V = \{ v_p(o) \mid p = 1, 2, \cdots, P \}$
- $E = \{(v_p(o), v_q(o)) \mid v_p(o) \in V, v_q(o) \in V, p \neq q, v_q(o) = \delta(v_p(o))\}$, 여기서 δ는 버전 생성함수이며, $v_q(o) = \delta(v_p(o))$는 $v_q(o)$가 $v_p(o)$에 대한 수정의 결과로 발생되었음을 의미한다.

버전 그래프에서 임의의 두 버전 $v_q(o)$과 $v_p(o)$는 둘 사이에 경로가 존재하느냐에 따라 리비전관계와 배리언트관계 둘 중의 하나를 가진다. 즉, $\{(v_q(o), v_k(o)), (v_k(o), v_{k+1}(o)), (v_{k+1}(o), v_{k+2}(o)), \ldots, (v_{k+n}(o), v_p(o))\} \subset E$이면 두 버전은 리비전관계에 있으며, 그렇지 않다면 배리언트관계에 있다. 프로세스가 수행되는 동안 문서의 변경은 이러한 리비전 및 배리언트관계를 기반으로 관리된다. 만일 프로세스의 단위업무들이 직렬로 연결되어 있다면, 이 때 사용되는 문서의 변경은 리비전으로 관리되며 병렬로 연결되어 있다면 배리언트로 관리된다.

실제 단위업무에서 이루어지는 변경 작업은 구조 연결모델에서 정한 바에 의하여 작업단위를 대상으로 이루어지며, 문서의 버전은 곧 작업단위들의 버전을 모아놓은 집합체가 된다. 이를 표현하기 위하여 '문서-작업단위간 연관(Document-Workunit Relation, DWR)'을 다음과 같이 정의한다.

정의 5. 문서-작업단위간 연관

문서-작업단위간 연관은 p-번째 문서 버전과 q-번째 작업단위 버전 간의 관계를 표현하기 위한 것으로 다음 세가지 유형 중 하나에 해당한다.

- $v_p(d) \leftrightarrow {}^{I}v_q(w)$는 d의 버전이 w의 버전

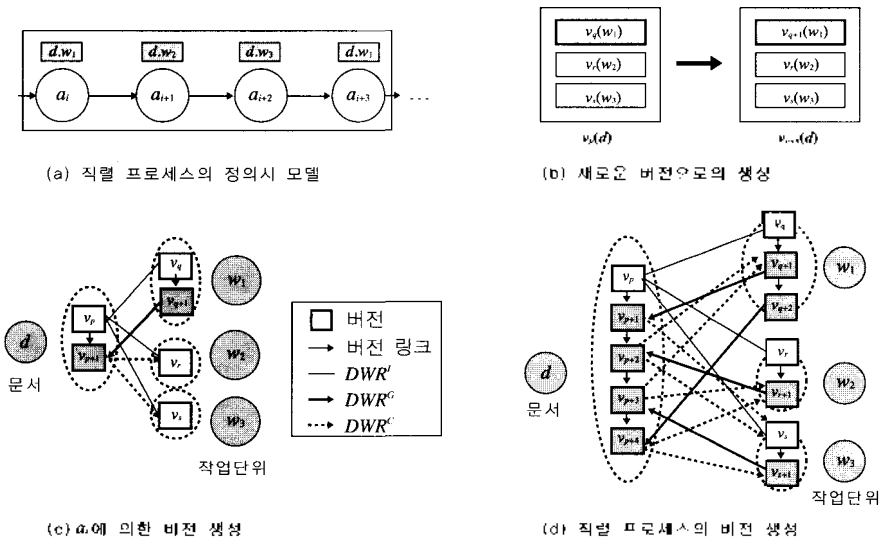
과 최초 연관이 있음을 의미하는 최초연관(DWR^I)이다.

- $v_p(d) \leftrightarrow {}^{G}v_q(w)$는 작업단위의 새로운 버전, $v_q(w)$이 생성되어 이 결과가 문서로 전파되어 문서 역시 새 버전, $v_p(d)$을 생성하게 한 생성연관(DWR^G)이다.
- $v_p(d) \leftrightarrow {}^{C}v_q(w)$는 문서 버전, $v_p(d)$이 내부적으로 작업단위 버전 $v_q(w)$을 가지고 구성되어 있다는 것을 의미하는 구성연관(DWR^C)이다.

DWR는 문서의 버전을 구성하는 데 유용하게 사용된다. 초기 문서 버전에 대해서는 DWR^I를 이용해서 구성하며, 프로세스 진행 중, 혹은 그 이후의 문서 버전에 대해서는 DWR^G과 DWR^C를 이용하여 구성한다. 본 연구에서는 DWR을 이용하여 프로세스의 수행 동안 발생하는 문서의 변경을 설명하기 위하여 프로세스의 가장 일반적인 구조인 직렬, AND-병렬, OR-병렬에 대하여 설명하기로 한다.

5.1.1 직렬 프로세스

직렬 프로세스는 모든 단위 업무들이 순차적으로 연결된 프로세스를 말한다. 이러한 직렬 프로세스의 경우, 문서의 버전은 항상 리비전이 생성이 된다. 다음 그림 5(a)는 직렬 프로세스의 간단한 예이다. 예제 프로세스에서 단위업무 a_i가 막 작업을 끝내었을 때, 새로이 생성된 버전을 그림 5(b)에 나타내었다. 다시 말해서, a_i는 w_1에 대한 작업을 통하여 새로운 문서 버전 $v_{p+1}(d)$을 만들게 된다. 이 과정을 더욱 자세히 설명하면, 그림 5(c)와 같다. 먼저, w_1에 대한 작업이 이루어졌으므로 $v_{q+1}(w_1)$을 만들고 w_1의 버전 그래프에 $(v_q(w_1), v_{q+1}(w_1))$링크를 추가한다. 이 때, w_1



(a) 직렬 프로세스의 정의시 모델

(b) 새로운 버전으로의 생성

(c) a_i에 의한 버전 생성

(d) 직렬 프로세스의 버전 생성

〈그림 5〉 직렬 프로세스에서의 변경관리

이 바뀐 것은 곧 문서 d가 바뀐 것을 의미하므로 새로운 문서 버전 $v_{p+1}(d)$이 생성되고 문서 d의 버전 그래프에 링크 ($v_p(d)$, $v_{p+1}(d)$)가 추가된다. 또한, $v_{p+1}(d)$와 $v_{q+1}(w_1)$ 사이에는 DWR^O이 만들어지며, $v_{p+1}(d)$와 나머지 작업단위버전 $v_r(w_2)$, $v_s(w_3)$과는 DWR^C이 형성된다.

프로세스의 각 단위업무가 수행되는 동안 위의 과정이 비슷한 방식으로 반복적으로 수행되며, 전체 프로세스가 완료되고 난 후의 변경 관리모델의 결과는 그림 5(d)와 같다. 지금까지 설명한 직렬 프로세스에서의 문서 변경 관리는 본 연구가 제시한 DWR에 의하여 다음과 같이 표현된다.

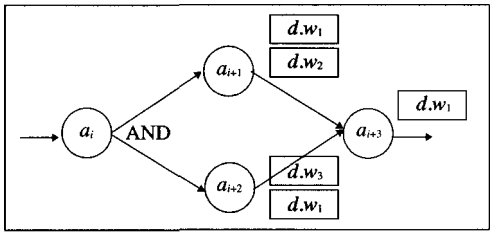
$$
\begin{aligned}
DWR = \{ \; & v_p(d) \leftrightarrow^{I} v_q(w_1),\ v_p(d) \leftrightarrow^{I} v_r(w_2),\ v_p(d) \leftrightarrow^{I} v_s(w_3), \\
& v_{p+1}(d) \leftarrow^{O} v_{q+1}(w_1),\ v_{p+1}(d) \rightarrow^{C} v_r(w_2),\ v_{p+1}(d) \rightarrow^{C} v_s(w_3), \\
& v_{p+2}(d) \rightarrow^{C} v_{q+1}(w_1),\ v_{p+2}(d) \leftarrow^{O} v_{r+1}(w_2),\ v_{p+2}(d) \rightarrow^{C} v_s(w_3), \\
& v_{p+3}(d) \rightarrow^{C} v_{q+1}(w_1),\ v_{p+3}(d) \rightarrow^{C} v_{r+1}(w_2),\ v_{p+3}(d) \leftarrow^{O} v_{s+1}(w_3), \\
& v_{p+4}(d) \leftarrow^{O} v_{q+2}(w_1),\ v_{p+4}(d) \rightarrow^{C} v_{r+1}(w_2),\ v_{p+4}(d) \rightarrow^{C} v_{s+1}(w_3) \quad \}
\end{aligned}
$$

5.1.2 AND-병렬 프로세스

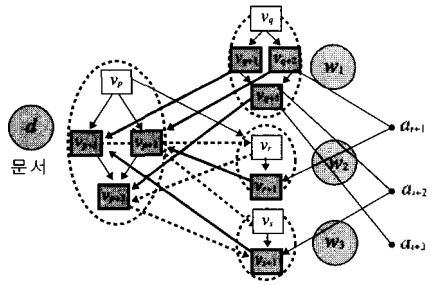
AND-병렬 프로세스는 단위업무를 병렬로 처리함으로써 작업의 효율을 높여주는 프로세스 구조이며, 문서작업 측면에서는 같은 문서를 한꺼번에 처리하여 배리언트를 생성하게 한다. 본 연구에서는 두 가지 종류의 AND-병렬 프로세스를 고려하는데, 하나는 경쟁 분기구조이며 다른 하나는 협동 분기구조이다.

먼저 경쟁 분기구조는 하나의 작업단위를 병렬적인 두 단위업무가 경쟁적으로 작업한 후 병합시에 하나로 병합하는 형태이고, 협동 분기구조는 병렬적인 단위업무들이 각기 다른 작업단위를 작업하여 이들 각 결과가 문서에 자동으로 반영되도록 하는 것이다. 두 분기 구조의 자세한 동작 방식은 [1]를 참조하기 바란다. 이들 두 가지 형태가 혼합되어 사용되는 경우도 발생하는데 본 논문에서는 혼합된 경우를 통해 두 가지를 설명한다.

그림 6(a)에서 보는 바와 같이, w_1은 모든 분기에서 경쟁적으로 처리하여 병합 단위업무 a_{i+3}에서 병합하며, w_2와 w_3은 각각에 대해 분기 단위업무에서 작업하고 이들 결과는 다음 버전의 문서에 자동으로 반영된다. 따라서, w_1은 경쟁 분기의 대상이며, w_2와 w_3은 협동 분기의 대상이다. AND-병렬 프로세스에서는



(a) AND-병렬 프로세스의 정의시 모델



(b) AND-병렬 프로세스의 변경관리

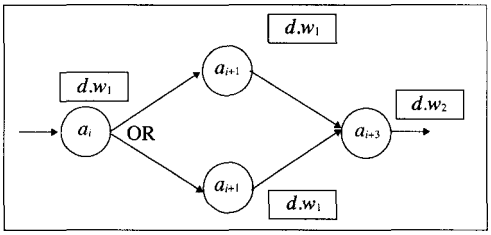
〈그림 6〉 AND-병렬 프로세스에서의 변경관리

직렬 프로세스와는 달리, 리비전뿐 아니라, 분기에 의한 배리언트가 형성되면서 버전이 자동으로 생성된다. AND-병렬 프로세스의 변경 관리모델은 그림 6(b)와 같다.

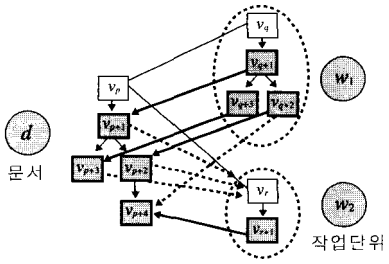
5.1.3 OR-병렬 프로세스

OR-병렬 프로세스는 여러 분기에서 작업한 결과 중 하나의 분기만이 의미를 지니게 되는 구조로 버전그래프에서도 하나의 분기에서 처리된 버전만이 계속 리비전을 허용한다.

다음 그림 7의 (a)와 (b)는 각각 OR병렬의 정의시 모델과 실행시 모델을 표현한 것으로 분기가 일어난 후 a_{i+1}, a_{i+2}에서 문서 d에 대한 작업을 처리한 후 a_{i+1}에 의해 처리된 $v_{p+2}(d)$ 버전만이 계속 리비전을 생성한다.



(a) OR 병렬 프로세스의 정의시 모델



(b) OR- 병렬 프로세스의 변경관리

〈그림 7〉 OR-병렬 프로세스의 버전 생성

5.2 변경 저장모델

변경 저장모델은 프로세스가 수행되는 동안 버전이 자동으로 생성되는 과정에서 버전 저장 공간의 낭비를 막기 위하여 작업에 의한 변경분을 자동으로 추출하고 이를 저장한다. 또한 이러한 변경분을 토대로 원하는 문서를 추출, 구성할 수 있도록 한다. 정의시에 단위업무가 작업해야 하는 작업단위가 정해져 있으므로, 시스템은 이를 통해 해당 작업단위가 어떻게 변경되었는지 추출할 수 있다. 추출된 작업단위에 대해서는 각 필드에 대하여 내용이 추가(ADD), 삭제(DEL), 수정(MOD)되었는지를 체크하여 변경을 저장한다. 단위업무에 의한 문서의 변경은 작업단위를 대상으로 이루어지며 다음과 같이 정의된다.

정의 6. 작업단위변경

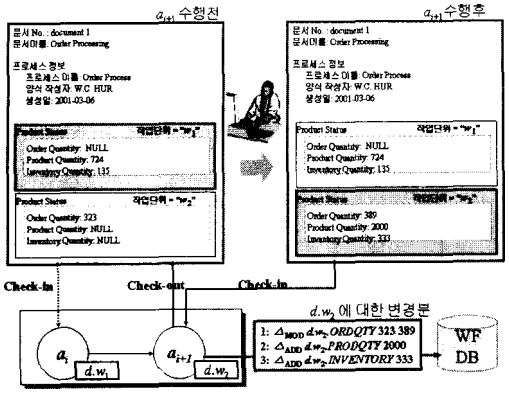
작업단위변경(δ)은 작업단위에 포함된 필드들의 변경 집합으로 정의된다.

$\delta = \{\triangle_l \mid l = 1, 2, \cdots, L\}$

이 때, L은 변경의 대상이 되는 작업단위에 속한 필드의 수가 된다. 또한, l-번째 필드에 대한 변경 $\triangle_l$은 다음의 네 가지 타입이 있다.

- 추가변경($\triangle_{ADD}$): 필드에 새로운 값이 추가되었음을 의미
- 삭제변경($\triangle_{DEL}$): 필드의 값이 지워졌음을 의미
- 수정변경($\triangle_{MOD}$): 필드의 값이 변경되었음을 의미
- 빈변경($\triangle_{\phi}$): 아무런 변경도 일어나지 않았음을 의미

문서의 저장과정은 그림 8과 같이 이루어



〈그림 8〉 문서 변경의 추출 및 저장

진다. 그림에서 보는 바와 같이 그림의 예제 프로세스에서 구조연관모델에 의하여 a_{i+1}에서 w_2에 대한 작업을 수행하므로($p.a_{i+1} \uparrow {}^{W}_{A} * = \{d.w_2\}$) w_2에 소속된 필드에 대한 변경을 기록하여 저장한다. 현재, w_2에는 세 개의 필드, Order Quantity, Product Quantity, Inventory Quantity가 있으며, 각각은 a_{i+1}에 의하여 내용이 수정 및 추가되었다. 이 때, 저장되는 변경의 내용은 그림 8의 아래쪽 오른편의 작은 박스와 같다.

이러한 변경에 대한 저장은 단위업무가 작업을 수행할 때마다 이루어지며, 프로세스의 수행과정은 초기 문서에 대한 변경의 반복적인 적용이 된다. 따라서, 각 필드는 프로세스가 수행되는 동안 추가, 수정, 삭제의 과정이 반복적으로 적용되어 최종적인 모습을 가지게 된다. 이 때, 특정 시점에서 문서를 사용자에게 보여주기 위해서는 문서의 초기 단계로부터 변경을 차례대로 적용할 수 있다. 이러한 과정은 변경 간의 결합을 통하여 정의할 수 있으며, 본 논문에서는 다음과 같은 변경의 결합을 정의한다.

정의 7. 변경의 결합

하나의 작업단위에 행해지는 연속적인 서로 다른 다수의 변경은 버전 재생성의 효율을 위하여 결합된 하나의 변경(δ)으로 표현할 수 있으며 이는 다음과 같이 표현된다.

$\delta = \delta_i \circ \delta_j = \{ \triangle_{il} \circ \triangle_{jl} \mid l = 1, \cdots L \}$

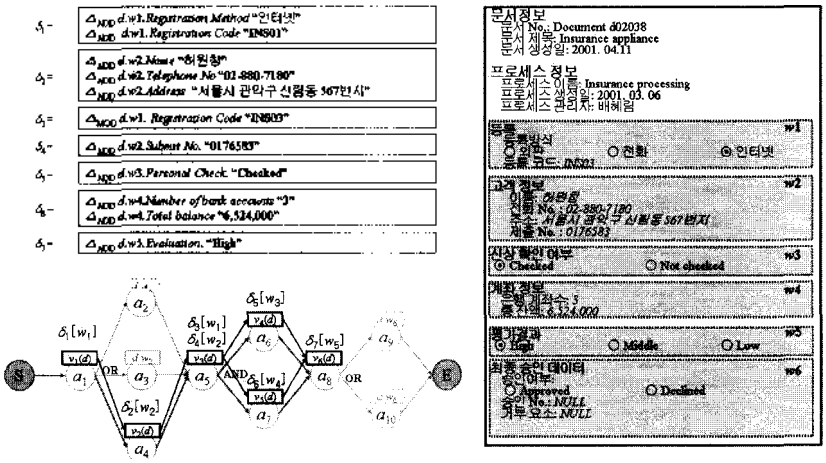
($\delta_i = \{\triangle_{i1}, \triangle_{i2}, \ldots, \triangle_{il}, \ldots, \triangle_{iL}\}$, $\delta_j = \{\triangle_{j1}, \triangle_{j2}, \ldots, \triangle_{jl}, \ldots, \triangle_{jL}\}$, 단, L은 작업단위에 포함된 필드의 수)

$\triangle_l \circ \triangle_m$의 계산은 본 논문의 부록에서 정의한 필드간 변경결합 연산을 따른다.

변경의 반복적인 적용은 프로세스의 특정 시점에서의 문서의 상태를 결정하게 되며, 프로세스가 수행되는 도중, 혹은 프로세스 완료 후에 특정 시점의 문서를 구성할 필요가 있다. 본 논문에서는 모든 문서의 버전을 저장하지 않고 최초 문서와 각 단위업무가 문서에 수행한 변경만을 저장하여 공간을 효율화하고 반복적인 변경의 결합을 하나의 변경으로 표현함으로써 문서를 효과적으로 구성하게 한다. 예를 들면, a_8가 작업을 끝내 시점의 문서d는 v_6가 되며, 변경간의 결합을 통하여 다음과 같이 구하여 진다.

$$\begin{aligned} v_6(d) &= (\delta_1 \circ \delta_2 \circ \delta_3 \circ \delta_4 \circ \delta_5 \circ \delta_7)\, v_0(d) \\ &= (\delta_1 \circ \delta_2 \circ \delta_3 \circ \delta_4 \circ \delta_6 \circ \delta_7)\, v_0(d) \\ &= \delta(v_0(d)) \end{aligned}$$

(단, δ = {$\triangle_{ADD}$ $d.w_1$.Registration Method "인터넷", $\triangle_{ADD}$ $d.w_1$.Registration Code "INS03", $\triangle_{ADD}$ $d.w_2$.Name "허원창", $\triangle_{ADD}$ $d.w_2$.Telephone No "02-880-7180", $\triangle_{ADD}$ $d.w_2$.Address "서울시 관악구 신림동 567번지", $\triangle_{ADD}$ $d.w_2$.Submit No. "0176583", $\triangle_{ADD}$



(a) 문서 d에 대한 변경 저장 기록

(b) a_8 이 끝난 후의 문서 버전 $v_5(d)$

〈그림 9〉 변경 정보를 이용한 문서의 재구성

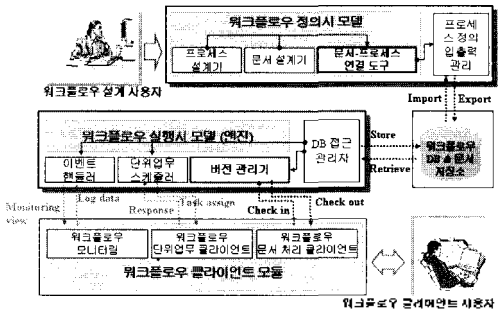
$d.w_3$.Personal Check. "Checked", $\triangle_{ADD}$ $d.w_4$.Number of bank accounts "3", $\triangle_{ADD}$ $d.w_4$.Total balance "6,524,000", $\triangle_{ADD}$ $d.w_5$.Evaluation. "High"})

6. 구현 및 시스템 구조

본 연구의 본문에서 제안한 바와 같은, 프로세스의 구조를 반영하여 자동으로 버전을 관리해주고 이를 저장 및 추출해주는 모델로 구성된 프레임워크에 대하여 프로토타입 시스템을 구현하였다. 이러한 시스템의 전체 구조는 그림 10과 같다.

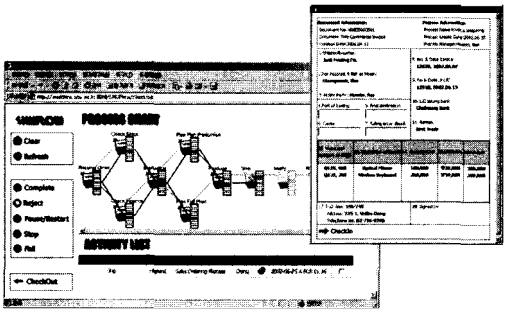
그림에서 보는 바와 같이 전체 시스템은 크게 정의시 모듈집합과 실행시 모듈집합, 그리고 클라이언트 모듈집합으로 구성된다. 먼저 정의시 모듈집합은 프로세스와 문서의 설계를 각각 담당하는 프로세스 설계기와 문서설계기가 있으며 문서-프로세스 연결도구를 이용하여 둘 사이의 연관을 맺은 후 저장한다.

실행시 모듈집합은 정의시 모듈에서 제공하는 모델을 바탕으로 프로세스를 실행하는데, 먼저 엔진은 프로세스의 진행을 조정, 제어, 통제한다. 본 연구에서는 서울대학교 산업공학과에서 개발한 SNUFlow 워크플로우 엔진을 사용하였다. SNUFlow엔진에 본 연구의 변경 관리모델을 구현한 버전관리기를 추가



〈그림 10〉 A전체 시스템 구조

함으로써 프로세스가 수행되는 동안 변경이 관리되도록 하였다.



〈그림 11〉 문서 변경관리를 위한 클라이언트 화면

클라이언트 모듈 집합은 프로세스의 진행 과정을 모니터링하고, 실제 업무가 할당되었을 때 문서처리를 병행하면서 단위업무를 수행하도록 한다. 사용자는 주로 워크플로우 클라이언트 모듈을 통해서 시스템과 상호작용하게 되는데 워크플로우 클라이언트 모듈의 주화면을 그림 11에 나타내었다. 그림에서 보는 바와 같이 각 단위업무의 상태를 확인하면서 작업을 수행하도록 프로세스 차트를 제공한다. 어두운 아이콘은 이미 작업이 완료된 단위업무를 표현하며, 내부에 화살표 원을 포함한 아이콘은 현재 진행중인 단위업무를 표현한다. 사용자는 아래쪽의 리스트를 확인하여서 자신에게 주어진 문서 관련 업무를 수행할 수 있다. 단위업무를 수행하기 위하여 사용자는 할당된 문서 버튼을 클릭하며, 시스템은 자동으로 사용자가 작업할 수 있는 형태의 문서를 본 논문의 모델을 기반으로 구성하여 체크아웃 해준다. 사용자는 오른편 상단과 같은 문서를 전달받아 작업을 하고 수정된 문서를 체크아웃 버튼을 눌러서 시스템에 전달한다. 시스템은 변경이 일어난 부분을 확인하여 변경분을 저장한다. 버전관리 시스템에서 사용자가 효과적으로 프로세스와 문서작업을 연계하도록 해당 단위업무위에 문서의 버전을 겹쳐 표현하였고 이는 본 논문의 모델에 의하여 문서의 변경이 정확하게 생성, 관리된다는 것을 보여주고 있다.

7. 결 론

본 논문에서는 프로세스 정보를 바탕으로 프로세스가 수행되는 동안 프로세스의 의미를 반영하여 자동으로 버전을 생성하고 이들을 관리하는 워크플로우 관리시스템의 새로운 프레임워크를 제시하였다. 이를 위하여 프로세스의 정의시에 문서를 프로세스 수행에 적합한 형태로 구조화하여 이를 프로세스 단위업무와 연계시키는 모델을 제안하였고, 이를 근거로 프로세스의 다양한 구조에 따라 문서의 버전이 자동으로 생성될 수 있음을 보였다. 또한, 각 버전이 효율적으로 저장 및 추출될 수 있도록 하는 모델로 제시하였다. 제안된 방법론은 기업에서 정형화된 양식문서를 기반으로 진행되는 프로세스에 적용 가능할 것으로 기대된다.

프로세스 정보를 바탕으로 한 문서 버전의 생성 및 관리는 여러 가지 장점을 제공한다. 먼저, 워크플로우 사용자가 작업해야 할 문서를 프로세스의 구조와 패턴에 맞게 자동으로 전달하여 줌으로써 사용자가 오류 없이 문서 작업을 할 수 있도록 도와준다. 또한, 문서 버전을 프로세스와 연계함으로써 문서의 변경

과정에 대한 사용자의 이해를 도와준다. 그리고, 프로세스의 전과정에서 문서를 추적하게 함으로써 프로세스가 종료되고 난 후에도 관리자가 특정 프로세스 위치에서 작업된 문서의 상태를 검색하는 것을 가능하게 한다. 특히, 여러 부서 혹은 기업이 관여된 복잡한 e-Business프로세스에서 문서를 포함한 다양한 자원의 변경관리에 효과적으로 적용될 것으로 기대한다.

본 연구에 대한 추후 연구과제로는 먼저, 문서 버전을 바탕으로 컨피규레이션[10]을 구성하고 이에 대한 버전관리를 수행하는 것을 들 수 있다. 이러한 컨피규레이션은 사용자가 원하는 조건의 문서 집합을 반환함으로써 효과적인 검색을 가능하게 한다. 또한, 본 논문의 버전관리의 대상인 문서뿐 아니라, 프로세스 정의 자체에 대한 버전을 관리하여 프로세스의 변경이력을 관리하는 것이 또 하나의 추후과제가 될 수 있다.

참 고 문 헌

[1] 배혜림, "e-Business 문서관리: 워크플로우 프로세스를 위한 비즈니스 문서 변경관리", 공학박사학위논문, 서울대학교, 2002.08

[2] I. B. Arpinar, U. Halici, S. Arpinar and A. Dogac, "Formalization of Workflows and Correctness Issues in the Presence of Concurrency," Distributed and Parallel Database, vol. 7, pp.199-248, 1999.

[3] H. Bae and Y. Kim, "A document-process association model for workflow management," Computers in Industry, vol. 47, no. 2, pp 139-154, 2002.

[4] R. Conradi and B. Westfechtel, "Version Models for Software Configuration Management," ACM Computing Survey, vol. 30, no. 2, June 1998.

[5] K. R. Dittrich and R. A. Lori, "Version Support for Engineering Database Systems," IEEE Transactions on Software Engineering, vol. 14, no. 4, pp. 429-437, Apr. 1988.

[6] A. Elmagarmid and W. Du, "Workflow Management: State of the Art Versus State of the Products", Proceedings of the NATO Advanced Study Institute on Workflow Management Systems (WFMS), Istanbul, Turkey, August 12-21, 1997.

[7] R. H. Guting, R. Zicari and D. M. Choy, "An Algebra for Structured Office Documents," ACM Transactions on Office Information Systems, vol. 7, no. 4, pp. 123-156, Apr. 1989.

[8] D. Hollingsworth, "Workflow management coalition specification: the workflow reference model", WfMC specification, 1994.

[9] Randy H. Katz, "Toward a Unified Framework for Version Modeling in Engineering Database," ACM Computing Surveys, vol. 22, no. 4, pp. 375-408, Dec. 1990.

[10] J. C. Miles, W. A. Gray, T. W. Carnduff, I. Santoyridis and A. Faulconbridge, "Versioning and configuration management in design using CAD and complex wrapped objects," Artificial Intelligence Engineering, vol. 14, no. 3, pp. 249-260, July 2000.

[11] E. Sciore, "Versioning and Configuration Management in an Object-Oriented Data Model," VLDB Journal, vol. 3, pp. 77-106, 1994.

[12] S. Y. W. Su, S. J. Hyun and H. M. Chen, "Temporal Association Algebra: A Mathematical Foundation for Processing Object-Oriented Temporal Database," IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 3, pp. 389-408, May/June 1998.

[13] L. Yu and D.J. Rosenkrantz, "A Linear-time Scheme for Version Reconstruction," ACM Transactions on Programming Languages and Systems, vol. 16, no. 3, pp 775-797, May 1994.

[14] A. Zeller and G. Snelting, "Unified Versioning through Feature Logic," ACM Transactions on Software Engineering and Methodology, vol. 6, no. 4, pp. 398-441, Oct. 1997.

부 록

A. 변경의 결합 방법

1. 역변경

단위 변경 △에 대한 역변경은 △-1로 표현하며 각 단위변경에 대한 역변경은 다음과 같이 정의된다.

- 추가변경의 역변경: $\triangle_{ADD}^{-1}=\triangle_{DEL}$

Ex: $(\triangle_{ADD}\ d_1.w_1.f_1$ "Yes"$)^{-1} = (\triangle_{DEL}\ d_1.w_1.f_1$ "Yes"$)$

- 삭제변경의 역변경: $\triangle_{DEL}^{-1}=\triangle_{ADD}$

Ex: $(\triangle_{DEL}\ d_1.w_1.f_1$ "Yes"$)^{-1} = (\triangle_{ADD}\ d_1.w_1.f_1$ "Yes"$)$

- 수정변경의 역변경: $\triangle_{MOD}^{-1}=\triangle_{MOD}'$

Ex: $(\triangle_{MOD}\ d_1.w_1.f_1$ "Yes" "No"$)^{-1} = (\triangle_{MOD}\ d_1.w_1.f_1$ "No" "Yes"$)$

2. 변경결합

하나의 필드에 대해서 이루어지는 순차적인 변경의 결과를 반영하기 위하여 단위 변경 간에 결합 연산을 정의한다. 이는 추후 문서의 재구성을 빠르게 수행할 수 있도록 하기 위함이다. 같은 필드를 대상으로 하는 두 단위변경△, △′ 사이의 결합 연산, △ ∘△′ 는 대상 필드에 대하여 △ 변경이 있은 후에 △′ 변경이 추가로 이루어졌음을 의미한다. 각 타입별로 가능한 단위변경간 결합 연산은 다음과 같이 정의된다.

- Case 1: $\triangle_{ADD} \circ \triangle_{ADD}' = \triangle_{ADD}'$

Ex: $(\triangle_{ADD}\ d_1.w_1.f_1$ "Internet"$) \circ (\triangle_{ADD}\ d_1.w_1.f_1$ "Phone"$) = (\triangle_{ADD}\ d_1.w_1.f_1$ "Phone"$)$

- Case 2: $\triangle_{ADD} \circ \triangle_{DEL} = \triangle_{\emptyset}$

Ex: $(\triangle_{ADD}\ d_1.w_1.f_1$ "Internet"$) \circ (\triangle_{DEL}\ d_1.w_1.f_1$ "Internet"$) = \triangle\emptyset$

- Case 3: $\triangle_{ADD} \circ \triangle_{MOD} = \triangle_{ADD}'$

Ex: $(\triangle_{ADD}\ d_1.w_1.f_1$ "Internet"$) \circ (\triangle_{MOD}\ d_1.w_1.f_1$ "Internet" "Phone"$) =$

$(\triangle_{ADD}\ d_1.w_1.f_1$ "Phone"$)$

- Case 4: $\triangle_{DEL} \circ \triangle_{ADD} = \triangle_{MOD}$

Ex: $(\triangle_{DEL}\ d_1.w_1.f_1$ "Phone"$) \circ (\triangle_{ADD}\ d_1.w_1.f_1$ "Internet"$) =$

(MOD $d_1.w_1.f_1$ "Phone" "Internet")

- Case 5: $\triangle_{MOD} \circ \triangle_{MOD}' = \triangle_{MOD}''$

Ex: ($\triangle_{MOD}$ $d_1.w_1.f_1$ "Phone" "Internet") $\circ$ ($\triangle_{MOD}$ $d_1.w_1.f_1$ "Internet" "e-mail") = ($\triangle_{MOD}$ $d_1.w_1.f_1$ "Phone" "e-mail")

- Case 6: $\triangle_{MOD} \circ \triangle_{DEL} = \triangle_{DEL}'$

Ex: ($\triangle_{MOD}$ $d_1.w_1.f_1$ "Phone" "Internet") $\circ$ ($\triangle_{DEL}$ $d_1.w_1.f_1$ "Internet") = ($\triangle_{DEL}$ $d_1.w_1.f_1$ "Phone")

- Case 7: $\triangle_{MOD} \circ \triangle_{ADD} = \triangle_{ADD}$

Ex: ($\triangle_{MOD}$ $d_1.w_1.f_1$ "Phone" "Internet") $\circ$ ($\triangle_{ADD}$ $d_1.w_1.f_1$ "e-mail") = ($\triangle_{ADD}$ $d_1.w_1.f_1$ "e-mail")

3. 변경법칙

위에서 제시한 각 연산들을 적용하는데 있어서 다음과 같은 법칙들이 적용된다.

- 빈변경은 변경에 아무런 영향도 주지 않는다.
  $\triangle_l \circ \triangle_\emptyset = \triangle_\emptyset \circ \triangle_l = \triangle_l$
- 두 변경에 대한 결합연산의 역연산은 각 변경의 역연산을 반대 순서로 적용한 것과 같다.
  $(\triangle_l \circ \triangle_l')^{-1} = \triangle_l'^{-1} \circ \triangle_l^{-1}$
- 하나의 변경과 그 변경의 역변경의 결합 연산은 아무런 변경을 가하지 않은 결과를 낳는다.
  $\triangle_l \circ \triangle_l^{-1} = \triangle_l^{-1} \circ \triangle_l = \triangle_\emptyset$
- 연산의 교환 법칙은 성립하지 않는다.
  $\triangle_l \circ \triangle_l' \neq \triangle_l' \circ \triangle_l$
- 연산의 결합 법칙이 성립한다.
  $(\triangle_l \circ \triangle_l') \circ \triangle_l'' = \triangle_l \circ (\triangle_l' \circ \triangle_l'')$

저 자 소 개

배혜림 (E-mail: hrbae@dongeui.ac.kr)
1996. 02 서울대학교 산업공학과(학사)
1998. 02 서울대학교 산업공학과(석사)
2002. 08 서울대학교 산업공학과 박사
2002. 08 ~ 2003. 02 삼성카드 정보기획팀
2003. 03 ~ 현재 동의대학교 인터넷비즈니스학과 전임강사
관심 분야 : 워크플로우 관리시스템, 다조직 프로세스 통합 방법론, XML, eAI

조재균 (E-mail : jgcho@dongeui.ac.kr)
연세대학교 응용통계학과(학사)
한국과학기술원 산업공학과(석사)
한국과학기술원 산업공학과 박사
1987 ~ 1998. 한국전자통신연구원 선임연구원
현재 동의대학교 경영정보 · 인터넷비즈니스학부 조교수
관심 분야 : 프로젝트 관리, 웹마이닝, 인터넷비지니스 모델 및 전략

정석찬 (E-mail: scjcong@dongeui.ac.kr)
1983 ~ 1987년 부산대학교 기계공학과(학사)
1987 ~ 1993년 일본 오사카부립대학 경영공학과 공학석사, 공학박사
1993 ~ 1997년 한국전자통신연구원 전자상거래연구부 선임연구원
1999 ~ 현재 동의대학교 인터넷비즈니스학과 조교수
관심 분야 : B2B전자상거래, SCM, ERP, 정보시스템

박기남 (E-mail: knpark@dongeui.ac.kr)
한국외국어대학교(문학사)
한국외국어대학교 경영학(석사)
경희대학교 경영학박사
한국정보과학연구센터 선임연구원
경인여자대학 인터넷비즈니스학과 전임강사
현재 동의대학교 인터넷비즈니스학과 전임강사
관심 분야 : 신기술가치평가, 마케팅정보시스템, 정보시스템 보안 및 감사, XBRL