실행시간캡슐화를 통한 워크플로우관리시스템의 구축
Development of Workflow Management System Using Run-time Encapsulation

정재윤, 김동수, 김영호, 강석호
서울대학교 산업공학과

Abstract - Workflow Management System(WfMS) enables corporations to manage business processes efficiently. The purpose of this research is to develop a management system that is capable of managing complicated business processes efficiently under distributed environment where many companies or divisions are participating. Run-time encapsulation proposed in this paper is a system design methodology that enables appropriate response to the dynamically changing processes by using nested processes. We have implemented WfMS under Web environment using run-time encapsulation. Run-time encapsulation using nested process is an efficient development methodology for implementing heterogeneous and distributed WfMS under web environment.

1. 서 론

워크플로우는 컴퓨터에 의해서 자동으로 실행되고 관리되는 업무프로세스이다. 그리고 워크플로우관리시스템은 워크플로우를 정의하여 실행시키며, 이를 관리할 수 있는 소프트웨어를 의미한다[1]. 워크플로우관리시스템은 최근에 기업활동을 지원하는 여러 가지 정보시스템들, 예를 들면, 전사적자원관리시스템(ERP: Enterprise Resource Planning), 제품정보관리시스템 (PDM: Product Data Management), 공급망관리시스템 (SCM: Supply Chain Management), 고객관계관리시스템 (CRM: Customer Relationship Management) 등의 기반이 되어 정형화된 업무를 수행하고 관리하는 중추적 역할을 수행하고 있다[2].

본 논문은 워크플로우관리시스템의 설계와 개발에 관한 방법론을 제시한다. 워크플로우관리시스템은 많은 기업에서 업무프로세스를 효율적으로 관리하기 위하여 이용되고 있다. 본 논문의 목적은 여러 기업이나 부서가 동시에 참여하여 분산된 환경에서 수행되는 복잡한 업무프로세스를 효율적으로 관리하는 시스템 개발하는 것이다. 중첩 프로세스(nested process)는 이러한 분산 기업환경의 복잡한 업무프로세스를 관리하기에 적합한 모델링 기법이다. 그리고, 실행시간캡슐화(run-time encapsulation)는 동적으로 변화하는 프로세스를 중첩프로세스를 사용하여 적절하게 대응할 수 있도록 한 시스템 설계방법이다. 본 논문에서는 실행시간캡슐화를 관리하기 위한 방법을 제시하고, 객체지향 관점에서 설계하여 워크플로우시스템을 효율적으로 관리하고 변경할 수 있도록 한다. 또한 워크플로우 관리 모듈을 컴포넌트화하여 다중서버로 확장 가능하게 한다.

본 논문은 다음과 같이 구성되었다. 2장에서는 논문의 기반이 되는 중첩프로세스 모델에 관하여 기술하였다. 3장에서는 실행시간캡슐화에 대해 설명하고, 이를 관리하기 위한 모델을 제시한다. 4장에서는 워크플로우의 객체화에 대해 기술하였다. 그리고 5장에서는 다중서버를 통한 시스템 구현 내용을 기술하였다. 그리고 6장에서는 결론 및 추후 연구과제를 서술하였다.

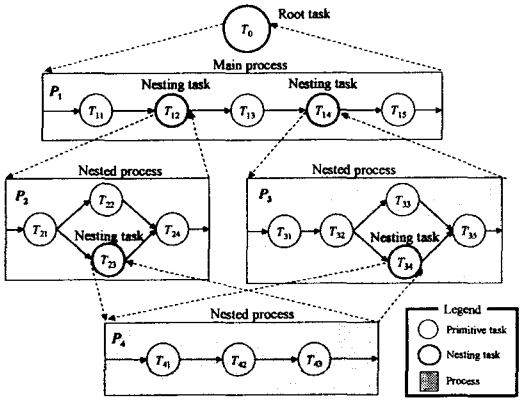
2. 중첩프로세스 모델

기업에서 수행되는 업무프로세스는 종종 복잡다단한 형태를 지닌다. 특히 분산환경에서 수행되는 프로세스는 여러 부서가 참여하고, 규모가 큰 복잡한 설계를 요구한다. 이러한 프로세스를 한꺼번에 설계하기 위해서는 모든 업무의 진행과정과 세부적인 업무내용을 파악

해야 하므로 상당한 노력과 어려움이 요구된다. 또한 한 수준에서 세부적인 단계까지 상세하게 설계된 프로세스는 이를 관리하고, 수정하는 데에도 효율적이지 못하다. 워크플로우관리시스템에서는 이처럼 복잡한 프로세스를 효율적으로 설계하고 관리하기 위하여 중첩프로세스 모델을 사용한다.

중첩프로세스 모델은 복잡한 프로세스를 수직적 단계로 나누어 설계할 수 있다. 상세한 설계는 하위프로세스를 통하여 표현함으로써 전체프로세스는 몇 개의 하위프로세스와 간단한 업무로 구성하게 된다. 즉 복잡한 프로세스의 설계가 하위프로세스에 의해 이들의 간단한 조합으로 구성할 수 있게 된다.

[그림 1]은 중첩프로세스 모델링의 예를 보여준다. 사각형은 프로세스를, 원형은 태스크를 나타낸다. 루트태스크(Root task) T_0으로 표현될 수 있는 상위프로세스(Main process) P_1은 하위프로세스(Nested process) P_2, P_3을 추상화하고 있는 중첩태스크(Nesting task) T_{12}, T_{14}와 업무수행을 위한 원시태스크(Primitive task) T_{11}, T_{13}, T_{15}로 구성된다. 그림의 최하위 프로세스 P_4는 두 상위프로세스 P_2, P_3에 동시에 중첩될 수도 있음을 보여주고 있다.



[그림 1] 중첩프로세스 모델[2]

3. 실행시간캡슐화

실행시간캡슐화(run-time encapsulation)는 중첩프로세스를 확장한 개념으로, 상위프로세스에 포함된 하위프로세스를 시스템 측면에서나 내용적 측면에서 상위프로세스와 명확히 분리함으로써 보다 동적이고 유연한 프로세스를 지원하는 것이 목적이다.

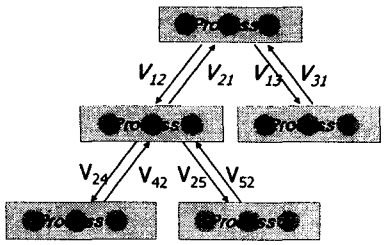
상위프로세스와 구분된 하위프로세스는 상위프로세스가 실행 중에 하위프로세스를 변경하거나 다른 하위프로세스로 대체하는 것이 가능해야 한다. 그래서, 상위프로세스의 실행 경과에 따라 하위프로세스를 적절하게 변경하거나, 다른 적합한 하위프로세스로 대체할 수 있고, 이외의 프로세스 요소들을 변경할 수 있어야 한다. 즉, 다음과 같은 기능을 수행할 수 있어야 한다.

- 하위프로세스의 실행을 다른 서버에서 분할 실행 가능.
  하나의 프로세스에 여러 부서나 조직이 참여한 다수의 하위프로세스가 존재할 때, 하위프로세스를 각 서버에서 자체적으로 수정하고 실행할 수 있고, 내용을 은닉할 수 있다. 각 서버에서 이미 실행중인 프로세스들을 임의로 조합하거나 새로운 통합프로세스로 발전시킬 수 있다.
- 프로세스 수행 도중 하위프로세스를 통하여 진행 과정이 변경 가능.
  상위프로세스와 분리되어 실행되는 하위프로세스는 상위프로세스가 진행중이더라도 프로세스 정의를 바꾸거나, 담당자를 변경하는 등 동적 변화가 가능하다.
- 프로세스 수행 전이나 수행 도중 하위프로세스를 교환 가능.
  프로세스를 실행하기 전이나 수행 도중에 상위 프로세스의 진행 경과에 따라서, 하위프로세스를 적합한 다른 프로세스로 교환하거나, 새롭게 설계할 수 있다.
- 상향식, 하향식 프로세스 디자인 가능.
  기존에 정의된, 또는 실행중인 프로세스를 묶어 나가는 방식의 상향식 프로세스 디자인이나, 개략적으로 정의된 프로세스에 하위프로세스를 세부적으로 채워나가는 하향식 프로세스 디자인이 가능하다.

3.1 프로세스 계층 트리

실행시간캡슐화는 프로세스의 중첩관계가 임의로 형성되고, 변경될 수 있으므로, 프로세스간의 관계를 관리하기 위한 특별한 모델이 요구된다. 본 논문에서는 중첩프로세스 구조에서 상위프로세스와 하위프로세스의 관계를 설정하기 위한 구조로 트리구조를 도입하였다. 중첩프로세스 구조는 프로세스의 상하위 관계를 명확하게 정의할 수 있다. 또한 실행시간캡슐화에서는 임의로 상하위프로세스가 추가, 변경, 삭제될 수 있기 때문에 이들 프로세스간의 관계를 관리, 유지하는 것이 특히 중요하다[3]. 이러한 프로세스들의 관리를 위해서 계층트리를 제안한다.

그리고, 프로세스의 상하관계를 나타내는 프로세스 계층 트리는 프로세스간의 업무흐름(control-flow)과 자료흐름(data-flow)에 접근하는 관리 권한을 설정하는데 사용할 수 있다. 트리의 노드를 통하여 프로세스 정의의 변경권한, 담당자의 변경 권한, 관리자의 통제권한, 관리자의 모니터링 권한, 데이터의 접근권한 등을 정의할 수 있다.



[그림 2] 프로세스 계층 트리

3.2 프로세스간 접근 권한

중첩프로세스간에 다른 프로세스의 관리나 데이터를 참조해야 하는 경우가 있다[4]. 이 때에 사용자나 관리자의 수준에 따라 정보의 공유정도를 제한하게 된다. 그리고 이는 프로세스에 따라서 다르게 정의될 수 있기 때문에 프로세스의 관계가 표현된 프로세스 계층 트리를 통해서 구체화하게 된다.

트리의 노드에는 여러 접근 권한을 나타내는 양방향 순서쌍을 저장한다. 순서쌍은 0과 1로 구성되며, 권한의 유무를 나타낸다. 프로세스간 노드에 저장된 순서쌍을 근거로 (1) 프로세스 정의 변경, (2)담당자 변경, (3)프로세스/태스크 통제, (4)모니터링, (5) 데이터 공유 등의 접근 권한을 판단할 수 있다. 이는 다단계로 접근을 관리할 수 있다.

[그림 2]의 예를 들어보자. *Process1*에서 *Process2*의 권한은 대응되는 노드에 표시된 V_{12}를 통해서 판정하게 된다. 즉, 순서쌍의 해당 원소의 값이 1일 때 권한을 가지게 된다. 이번에는 인접하지 않은 프로세스간의 접근 권한을 살펴보자. *Process4*에서 *Process3*으로의 권한은 위와 같은 방법으로, V_{42}, V_{21}, V_{13}을 순차적으로 탐색함으로써 판정할 수 있다.

4. 객체지향 워크플로우

객체지향의 중심개념은 객체(object)이다. 객체는 하나의 실체(entity)로서, 일반적으로 속성(attribute)과 메소드(method)로 정의된다. 속성이란 객체를 다른 객체와 구별되게 하는 내재적인 특징을 나타내고, 메소드란 객체가 어떻게 동작하는가를 규정한다[5].

워크플로우에서의 기본단위는 프로세스라 할 수 있다. 프로세스는 워크플로우에서 고려할 수 있는 추상화 단위로서, 태스크, 작업자, 실행상태 등의 속성값들과, 프로세스의 시작, 통제, 하위프로세스 관리 등의 메소드를 포함하게 된다.

4.1 워크플로우의 객체화

여기서는 워크플로우에서 객체를 형성하는 단위가 되는 프로세스를 어떻게 설계함으로써 객체지향적 특징을 추구할 수 있는지 알아본다.

프로세스정의의 확장성을 위해 추상프로세스형(APT; Abstract Process Type)을 제안한다. 다음 예를 살펴보자. 기업에서 어떤 제품을 구입하는 구매프로세스는 거시적으로는 공통적 업무흐름을 가지만, 제품특성의 차이에 의해 각기 다른 정형의 구매프로세스를 가지고 있다. 세부적인 차이에 따라서 각기 다른 프로세스로 정의하여 관리, 사용할 수도 있다. 하지만 공통적으로 수행되는 과정을 추상적 프로세스를 정의하고, 특정제품에 따라 프로세스에 업무를 추가하거나 세부과정을 변경하여 사용할 수 있을 것이다. 중첩프로세스를 사용하여 이러한 변화를 반영한다면 다양한 제품의 구매프로세스를 정의하고 수정하는데 더 효율적일 것이다.

이와 같은 방법으로 프로세스를 구현할 때 얻을 수 있는 객체지향적 특성은 다음과 같다.

- 추상화(abstraction)
  추상프로세스형(APT)을 통하여 개괄적인 프로세스를 정의하고 관리할 수 있다. 동일한 과정을 거치는 프로세스들을 상위수준에서 정의하여 공통적인 변경사항과 관리를 총괄할 수 있다.
- 은닉화(encapsulation)
  중첩프로세스 모델링을 통하여 하위프로세스를 은닉화함으로써 상위수준에서 개념적 이해와 관리가 용이하다. 여러 수준에서 독립적으로 설계할 수 있도록 한다.
- 상속성(inheritance)
  기존 프로세스 정의의 변경을 상속을 통하여 실현함으로써, 다양한 버전과 특성을 지닌 프로세스를 효율적으로 생성, 관리할 수 있다. 국소적으로 다양한 형태를 가지는 프로세스를 집합체인 상위프로세스라는 모체를 통하여 관리할 수 있다.
- 다형성(polymorphism)
  동일한 상위프로세스가 실행되더라도 하위프로세스가 어떻게 정의되어 실행되느냐에 따라서 상위프로세스는 다른 과정으로 진행된다.

4.2 프로세스의 컴포넌트화

워크플로우에서 프로세스 정의는 반복하여 사용된다. 워크플로우에서 다루고 있는 프로세스는 정형화된 업무프로세스로서 동일한 업무과정이 다양하게 사용될 수 있다. 그리고 실행시간캡슐화에서는 이러한 프로세스들이 임의로 분리되거나 대체사용될 수 있어야 하므로, 독립적인 역할을 수행할 수 있어야 한다. 이러한 기능을 수행하기 위해 프로세스를 컴포넌트화할 수 있다.

컴포넌트란 특정 표준에 맞게 설계된 것으로 같은 표준에 의해 설계된 다른 객체와

결합하여 기능적인 그룹을 이룰 수 있는 객체를 말한다. 정형화된 업무프로세스를 컴포넌트화함으로써 반복적인 사용이나 적절한 변경을 통하여 실행시간캡슐화에 효율적으로 이용할 수 있다. 복잡한 프로세스의 경우 여러 개의 태스크로 이루어져 있어서 이들을 관리상의 이유, 또는 업무성격의 차별성 등으로 임의로 하위 프로세스 형태로 컴포넌트화하여 다른 상위 프로세스에서 재사용 가능하도록 구성할 수 있다. 위와 같이 프로세스의 컴포넌트화를 위해서 다음과 같은 특성들을 보장하여야 한다.

- 가분성(separability)
  기존의 복잡한 프로세스에서 의미있는 하위 프로세스를 컴포넌트로 쉽게 분리할 수 있어야 한다.
- 독립성(independency)
  분리된 프로세스는 상위프로세스와 독립적으로 자신의 기능을 수행할 수 있어야 한다.
- 재사용성(reusability)
  독립된 프로세스는 개별적으로, 또는 다른 프로세스의 하위프로세스로 재사용 가능하여야 한다.
- 대체가능성(replaceability)
  기존의 하위프로세스 대신에 다른 하위프로세스를 임의로 대체가능하여야 한다.

5. 다중서버 시스템

위에서 언급한 실행시간캡슐화를 반영하여 워크플로우관리시스템을 다중서버에서 실행될 수 있도록 설계하였다. 다중서버로 확장가능한 워크플로우관리시스템은 다음과 같은 장점을 가진다. (1)확장성, (2)성능, (2)안정성, (3)보안성, (4)조직간 독립성 유지, (5)이기종 시스템을 위한 확장성, (6)여러 유형의 워크플로우 혼합. 우리는 두 개 이상의 서버에서 상호운용이 가능한 워크플로우관리시스템을 구현하였다. 이 시스템의 엔진은 이질적 환경을 지원하기 위해 자바로 구현하였고, 시스템의 클라이언트는 애플릿으로 구현하여 인터넷, 인트라넷을 통하여 쉽게 접근이 가능하도록 하였다.

6. 결 론

본 논문에서는 분산환경에서 수행되는 복잡한 업무프로세스를 설계하고 관리하기 위한 방법론인 실행시간캡슐화를 구체화하며, 객체지향워크플로우 관점에서 접근하였다. 이 방법론은 워크플로우의 다단계설계, 분산서버로 확장, 프로세스간 관리, 접근권한 등에서 많은 이점을 보여주었다. 특히 분산환경에서 여러 부서나 기업이 참여하는 협업과정을 프로세스로 설계하여 관리하는 경우에 더욱 그러하다.

중첩프로세스를 통한 실행시간캡슐화는 웹환경을 기반으로 한 이질적 분산환경에서 워크플로우관리시스템을 구축하는데 효과적인 개발방법이 될 것이다.

Acknowledge

본 연구는 한국과학재단 특정기초연구비(97-02-00-09-01-3)지원으로 수행되었으며 지원에 감사드립니다.

참 고 문 헌

[1] D. Hollingsworth, "Workflow Management Coalition Specification: The Workflow Reference Model," *WfMC specification*, 1994.

[2] Yeongho Kim, Suk-Ho Kang, Dongsoo Kim, Joonsoo Bae, Kyung-Joon Ju, "WW-FLOW: Web based workflow management with runtime encapsulation", *IEEE Internet Computing*, vol.4, May.2000, pp.55-64.

[3] A. Kumar, J.L. Zhao, "Dynamic Routing and Operational Controls in Workflow Management Systems", Management Science, vol.45, No.2, Feb.1999, pp.253-273.

[4] G.Q. Huang, J. Huang, K.L. Mak, "Agent-based workflow management in collaborative product development on the Internet", Computer-Aided Design, vol.32, Feb.2000, pp.133-144.

[5] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice Hall, 1991.