워크플로우 시스템에서 프로세스 템플릿 검색을 위한 API 설계 및 구현

김행이[0] 이승일 한동수
한국정보통신대학원대학교
{hykim, gaialee, dshan}@icu.ac.kr

Design and Implementation of the API for Retrieving Process Template in Workflow System

HangYi Kim[0] SeungIL Lee DongSoo Han
Information and Communication University

요 약

워크플로우 시스템에서 워크플로우 프로세스에 관한 정의 정보는 일반적으로 데이터베이스에 저장되고 워크플로우 엔진은 이 데이터베이스로부터 필요한 정보를 인출하여 사용하게 된다. 엔진이 데이터를 검색하는 방법은 데이터베이스의 구조와 엔진의 구조에 따라서 매우 다양하다. 그 방법 중 하나로 워크플로우 엔진이 프로세스 정의가 저장된 데이터베이스를 직접 접근하여 원하는 정보를 가져올 수도 있으나 엔진과 데이터베이스 사이에 인터페이스를 두고서 간접적으로 접근하는 것은 많은 이점을 제공한다. 본 논문에서는 이러한 접근 방식의 이점 및 그 구현 기법에 관하여 ICU/COWS 라는 분산 워크플로우 시스템을 대상으로 논의한다.

1. 서론

워크플로우 관리시스템은 비즈니스 업무를 컴퓨터에 의해 자동으로 실행하고 관리하는 소프트웨어 시스템이다. 비즈니스 업무의 자동화를 위해서는 비즈니스 프로세스를 분석, 모델화, 기술하고 문서화하는데 많은 다양한 도구들이 사용된다. 워크플로우 관리시스템은 이러한 도구들을 포함하고 있다. 그 동안 많은 워크플로우 시스템이 각 업체에 의하여 만들어지면서 하나의 워크플로우 시스템에서 구현된 워크플로우가 임의의 타 워크플로우 시스템에서도 구동할 필요성 등이 대두되어 WfMC (Workflow Management Coalition) 라는 워크플로우 시스템 표준화 단체가 구성되었다.

여러 가지 분야에서 표준을 제정하고 있는 WfMC 는 그 중에 워크플로우 정의에 엑세스하고 이것의 기술에 있어서 시스템 사이에 호환성을 제공하기위해 워크플로우 프로세스 정의 메타모델에 대한 표준을 마련하였다[1]. 이 메타모델은 프로세스 정의 내에서 일반적으로 사용되는 엔티티들을 표현하고 있다. WfMC가 제시한 워크플로우 프로세스 정의 메타모델은 프로세스 정의를 포함하는 엔티티를 유지하기위해 프로세스 정의 저장소의 사용을 전제로 한다. 이 저장소에 있는 정보는 나중에 워크플로우 관리시스템의 엔진에 의하여 읽히게 된다. 표준에서는 프로세스 정의 데이터를 저장소에 효율적으로 전달하기위해 워크플로우 모델을 도입하여 많은 프로세스 정의에 공통되는 데이터 엔티티를 그룹화함으로써, 이들이 각 프로세스 정의 내에서 재정의되는 것을 방지하였다. 워크플로우 관리시스템이 프로세스 정의 정보를 저장소로부터 이 워크플로우 모델에 기반하여 구성된 데이터를 가져오는 경우에 이들 정보에 관한 조작이 필요하다.

일반적으로 워크플로우 시스템에서 워크플로우 프로세스에 관한 정의 정보는 데이터베이스에 저장되고 워크플로우 엔진은 중간에 인터페이스를 두고서 간접적으로 접근하여 이 데이터베이스로부터 필요한 정보를 인출하여 사용하게 된다. 이때 엔진 내부에 데이터베이스를 접근하는 루틴을 내장할 수도 있으나 인터페이스를 사용함으로써 워크플로우 엔진이 효율적으로 정보를 검색할 수 있고 다양한 프로세스 정의 정보를 탐색할 수 있다. 본 논문에서는 이러한 접근 방식의 이점 및 그 구현 기법에 관하여 ICU/COWS(Information and Communications University/Connector Oriented Workflow System) 라는 분산 워크플로우 시스템을 대상으로 논의한다.

ICU/COWS 시스템은 Joint flow 의 명세 (specification) 에 기반하고 분산 객체 instance 를 task controller 로 활용하는 구조를 가진 워크플로우 시스템으로 ICU 소프트웨어시스템 연구실에서 개발한 워크플로우 관리시스템이다[2]. 한편 Joint flow 는 새로운 소프트웨어 구조를 연구하기위해 관련된 회사들이 컨소시엄 형태로 구성한 단체인 OMG (Object Management Group) 에서 선정된 워크플로우 시스템에 관한 표준안으로, WfMC 표준을 수용하면서 OMG 에 적합한 형태로 변형된 형태를 가지고 있다[6].

본 논문의 2 장에서는 ICU/COWS 워크플로우 데이터베이스 스키마를 소개하고 3 장에서는 ICU/COWS 시스템에서 워크플로우의 프로세스 템플릿 활용 위한 인터페이스에 대해 알아본다. 마지막으로 4 장에서는 본 논문의 결론과 향후 과제에 대하여 논한다.

2. ICU/COWS 워크플로우 데이터베이스 스키마

프로세스 정의는 다음과 같이 정의된다. 워크플로우 관리시스템에 의하여 모델링이나 enactment 같은 자동화된 조작을 지원하는 형태로 비즈니스 프로세스를 표현하는 것이다. 프로세스 정의는 단위업무의 네트워크와 이들간의 관계, 프로세스의 시작과 종료를 나타내는 규정, 참여자, 결합된 IT 애플리케이션과 데이터 등과 같은 개개의 단위업무에 대한 정보로 구성된다[3].

ICU/COWS 워크플로우 스키마에서 개개의 프로세스 정의와 프로세스 정의에 응용될 수 있는 모든 엔티티 데이터는 그룹화되고, 이 엔티티들은 개개의 프로세스 정의에서는 생략되었다.

따라서, 프로세스모델내에 포함된 각 프로세스 정의는 프로세스 모델로부터 공통된 속성을 자동으로 상속 받을 수 있다. ICU/COWS 워크플로우 시스템에서 워크플로우 프로세스에 대한 정의 정보는 관계형 데이터베이스(relational DB) 에 저장되어 있다. 그림 3.1. 은 ICU/COWS 워크플로우 데이터베이스 스키마를 보여준다.

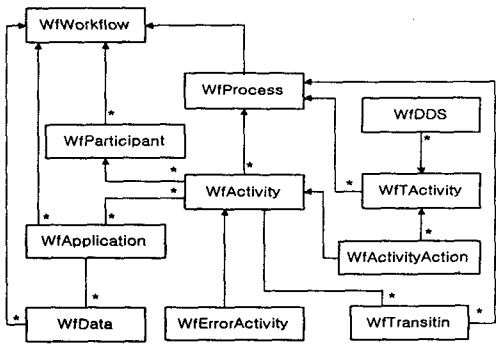

그림 3.1. ICU/COWS 워크플로우 데이터베이스 스키마

데이터베이스 스키마에서 중요한 테이블은 프로세스 단위업무의 자원(WfParticipant), 애플리케이션(WfApplication), relevant data (Wf Data) 와 이행(WfTransition) 이다. 이들은 WfWorkflow 나 WfProcess 의 일부분으로써, WfActivity 와 결합되어 구축되었다. 이 데이터베이스 스키마로부터 필요한 프로세스 단위업무에 관한 정보를 얻으려면 데이터베이스 스키마의 테이블간 연결관계를 고려하여 필요한 테이블을 결합하여야 한다는 것을 알 수 있다.

3. ICU/COWS 시스템의 프로세스 템플릿 활용을 위한 인터페이스

3.1. ICU/COWS 워크플로우 엔진의 프로세스 단위업무 정보 인출방법

워크플로우 프로세스 정의는 프로세스 자체를 기술하고 프로세스 정의 관리와 결합되거나 프로세스 실행 중에 프로세스 수준에 사용되는 옵션을 제공한다. 하나의 프로세스 정의는 하나 이상의 단위업무로 구성되어 있다. 하나의 단위업무는 자원과 컴퓨터 애플리케이션을 조합하여 진행될 수 있는 업무이다. 단위업무의 종류에 따라 relevant data나 이행정보가 필요하다. 프로세스 인스턴스는 프로세스 정의에 따라 워크플로우 관리시스템에서 생성, 관리 종료된다. 단위업무 인스턴스는 프로세스 정의에 따라서 프로세스 실행시에 요구되면 워크플로우 관리시스템에 의해서 생성되고 관리된다[1].

워크플로우 엔진은 워크플로우 인스턴스에 대하여 런타임 실행환경을 제공하는 소프트웨어이다[4]. ICU/COWS 시스템의 엔진을 구성하는 핵심 컴포넌트는 GTMIG(Global Task Manager Instance Generator), TMIF(Task Manager Instance Facto ry), GTMI(Global Task Manager Instance), TMI(Task Manager Instance) 이다. GTMIG 는 GTMI 와 TMI 의 생성을 제어하고 TMIF 는 GTMIG 와 TMI 를 생성하고 이들의 object factory 역할을 한다. TMI 는 GTMI 의 한 단위로 단위업무를 제어한다. GTMI 는 프로세스가 생성되고 생성된 프로세스가 소멸될 때까지 생성된 프로세스에 관련된 전반적인 사항을 관리하는 제어기이다[5]. GTMI 는 먼저 데이터베이스로부터 해당 프로세스에 대한 워크플로우 데이터베이스 스키마 를 읽어서 relevant data structure 를 구축하고 이 정보를 바탕으로 런타임에 필요한 데이터로 변환된 enhanced data structure 를 구축한다.

워크플로우 엔진이 데이터베이스로부터 필요한 정보를 검색할 때, 해당 데이터 테이블이 다른 테이블과 결합되거나 연결되어 있으므로 필요한 정보만을 검색하기 위한 조작이 필요하다. 워크플로우 엔진이 인터페이스를 경유하지 않고 직접 데이터베이스로부터 정보를 인출하는 방법은 다음과 같다.

- 해당 테이블의 정보를 데이터베이스로부터 검색한다.
- 별도로 해당 테이블의 정보를 데이터베이스로부터 검색하여 연결리스트(LinkedList) 로 연결한다.

위의 테이블정보와 연결리스트를 결합하여 필요한 정보를 검색한다. 이때, 두 가지 방법이 있다. 첫째, WfWorkflow 의 일부분이고 WfActivity 에 Foreign Key 가 존재하는 경우이다. 예를 들면, WfParticipant 정보를 얻을 때 이용하는 방법이다. WfActivity 테이블 정보 중 participant ID 와 model ID 를 매개변수로 하여 생성한 WfParticipant 연결리스트의 ID 를 비교하여 일치하는 WfParticipant의 정보를 검색한다. 둘째, WfWorkflow 의 일부분이고 WfActivity 에 Foreign Key 가 존재하지 않는 경우이다. 예를 들면, WfApplication 의 정보를 얻을 때 이용하는 방법이다. WfActivity 와 WfApplication 를 연결하는 새로운 테이블(Assign) 을 데이터베이스 스키마에 삽입하고, WfActivity ID 를 매개변수로 하여 Assign 연결리스트를 생성하고 model ID 를 매개변수로 하여 WfApplication 연결리스트를 생성한 후 이들의 application ID 가 일치하는 WfApplication 의 정보를 검색한다.

엔진 내에서 직접 정보를 인출할 경우 엔진이 복잡해지고 엔진과 데이터베이스의 의존성이 높아 유지 관리가 어렵다. 이러한 단점을 보완하기 위해서 그림 3.2. 와 같이 데이터베이스로부터 정보를 인출할 때 해당 데이터를 검색하기위한 인터페이스를 사용한다. 테이블정보 리스트는 테이블정보 연결리스트를 반환하고 테이블요약정보 리스트는 테이블요약정보 연결리스트를 반환한다.

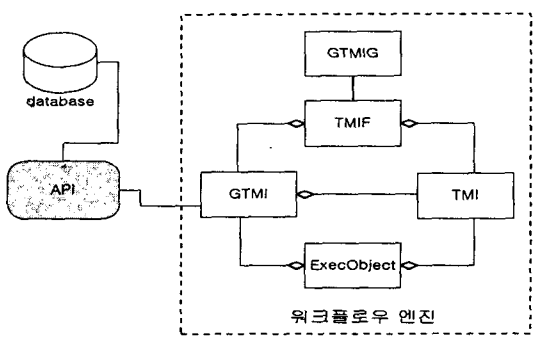

그림 3.2. ICU/COWS 워크플로우 엔진의 데이터 접근방법

3.2. 프로세스 템플릿 활용을 위한 인터페이스

그림 3.3. 은 데이터베이스 스키마의 테이블로부터 생성되는 인터페이스와 이들을 결합하여 생성되는 인터페이스의 대표적인 예를 나타내고 있다. 이를 위해 그림 3.1. 의 테이블간의 관계와 일부 테이블을 생략하였다. 예를 들면, 그림에서 WfWorkflow 테이블은 테이블정보(Workflow) 검색, 테이블요약정보(WorkflowInfo) 검색, 테이블정보 리스트(WorkflowList) 검색, 그리고 테이블요약정보 리스트(WorkflowInfoList) 검색을 위한 인터페이스를 갖는다. 이때, 요약정보는 해당 테이블과 연결된 테이블의 연결정보 필드를 제외한 것을 의미한다. 이 요약정보는 불필요한 필드를 제거함으로써 엔진의 로드를 감소시켜 효율을 증가 시킨다. 테이블정보는 테이블 객체를 반환하고 테이블요약정보는 테이블요약정보 객체를 반환한다.

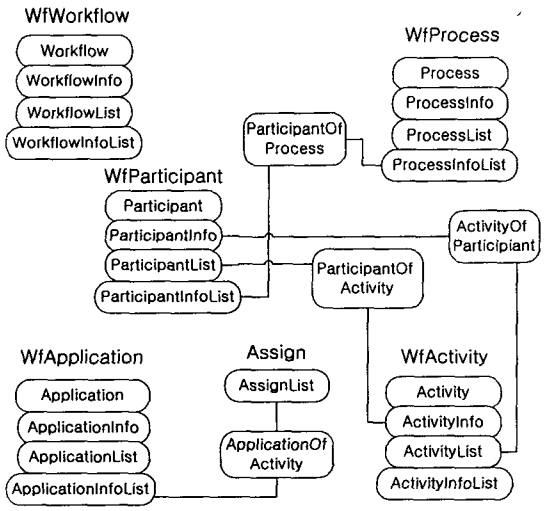

그림 3.3. 프로세스 템플릿 활용을 위한 인터페이스 구현 예. 굵은 글씨는 그림 3.1. 의 테이블을 의미하고 둥근 사각형은 인터페이스를 의미한다.

이 네 종류의 인터페이스는 단독으로 워크플로우 엔진에 사용되기도 하고 서로 결합하여 결합정보를 제공하는 인터페이스를 생성시키기도 한다. 주로 프로세스 단위업무 정보가 해당된다. 각 인터페이스의 매개변수에 따라서 다양한 결합방법이 이용된다. 그림 3.3. 에 테이블간의 인터페이스를 결합하는 대표적인 예가 나타나있다. 결합방법은 다음과 같다.

- 데이터베이스 스키마에서 같은 테이블을 매개로 연결되는 경우 (WfParticipant, WfProcess) 는 매개하는 테이블(WfWorkflow)의 ID 를 매개변수로 하여 각 테이블의 연결리스트 (ProcessList, ParticipantInfoList) 를 결합하여 연결된 테이블의 관련정보 (ParticipantOf - Process) 의 객체 연결리스트를 반환한다.
- 3.1. 에서 설명된 첫번째 방법처럼 연결된 테이블의 정보를 검색하는 경우는 정보를 요구하는 테이블의 요약정보 (ActivityInfo) 와 정보를 제공하는 테이블의 리스트 (ParticipantList) 를 결합하여 관련정보(Participant-OfActivity) 의 객체를 반환한다.
- 3.1. 에서 설명된 두번째 방법처럼 새로운 데이터테이블을 삽입하는 경우는 삽입된 테이블의 리스트 (AssignList) 를 만들고 이것과 정보를 제공하는 테이블의 리스트 (ApplicationtList) 를 결합하여 관련정보 (ApplicationOfActivity) 의 객체 연결리스트를 반환한다.
- 프로세스 엔티티가 관련되는 단위업무들의 정보를 검색하는 경우는 해당 엔티티의 요약정보 (ParticipantInfo) 의 ID 와 프로세스의 단위업무정보 리스트 (ActivityList) 를 결합하여 해당 단위업무 (ActivityOfParticipant) 의 객체 연결리스트를 반환한다.

위에서 제시한 인터페이스 결합방법은 데이터베이스 스키마에서 테이블간의 관계만 알면 모두 확대 적용할 수 있으므로 워크플로우 엔진이 요구하는 모든 인터페이스를 생성할 수 있다.

위에서 언급된 데이터베이스 스키마 테이블로부터 유래되는 인터페이스 이외에 워크플로우 엔진이 필요로 하는 워크플로우 정보는 프로세스 수 (int numberOfProcess), 프로세스의 워크플로우 소속여부 (Boolean isMemberFromWorkflow) 등을 지원하는 인터페이스가 있다. 현재 ICU/COWS 시스템에서 프로세스 템플릿 활용을 위한 인터페이스가 60 여 개 구현되어 있다. 이들은 워크플로우 엔진이 필요로 하는 모든 정보를 제공한다.

4. 결론 및 향후 연구과제

워크플로우 엔진이 인터페이스를 활용하는 방법은 워크플로우 엔진의 다른 기능과 관련되어 또 다시 확장되므로 이 부분은 제외하고, 이 논문에서는 데이터베이스 스키마 테이블간의 관계로부터 정보를 얻는 인터페이스를 만들고, 워크플로우 엔진은 단순히 이를 검색하여 사용하는 관점만을 고려하였다.

3.3.항에서 언급된 것과 같이 다양한 종류의 API 가 구현됨으로써, 워크플로우 엔진은 해당 프로세스의 요약정보, 해당 프로세스의 전체정보, 해당 프로세스를 구성하는 단위업무의 정보 등을 직접 결합하여 추출하지 않고 필요에 따라 해당 인터페이스만을 호출하도록 함으로써 시스템 구축이 단순화 되고 용이해졌다. 또한 워크플로우 템플릿이 변경되는 경우 주로 인터페이스를 변경함으로써 엔진에 미치는 영향이 감소되었다.

워크플로우 템플릿이 변경되고 이로 인하여 프로세스 인스턴스가 영향을 받게 되는 동적변경 환경에서 효과적으로 대응하는 인터페이스를 생성하고 이를 워크플로우 엔진에 반영하는 방법은 추후에도 계속 연구되어야 할 과제이다.

5. 참고 문헌

[1] Workflow Management Coalition Specification Document, " Interface 1: Process Definition Interchange Process Model" , Version 1.1, October 1999.

[2] Dongsoo Han, Jaeyong Shim, Chansu Yu , " ICU/COWS: A Distributed Transactional Workflow System Supporting Multiple Workflow Types " IEICE Transactions on Information and Systems Vol. E83-D, No. 7, July 2000

[3] Workflow Management Coalition Specification Document, " Terminology & Glossary" , Issue 3.0, February 1999.

[4] Workflow Management Coalition Specification Document, " The Workflow Reference Model " , Issue 1.1, January 1995.

[5] Seungll Lee, JaeYong Shim, DongSoo Han, " Using Design Patterns in the Development of Object -Oriented Workflow Management System Engine " ' 한국정보과학회 가을 학술발표논문집, 26(2):537-539, 1999.

[6] Object Management Group, Inc., " Workflow Management Facility" , Document Number: dtc/2000-02-05.