워크플로우 관리 시스템의 설계 및 구현

신 동 일† · 신 동 규††

요 약

워크플로우는 문서, 정보, 태스크가 한 사용자에서 다른 사용자로 일련의 업무절차 규칙에 의한 처리를 위해 전달되는 비즈니스 프로세스의 자동화를 의미하며, 워크플로우 관리시스템은 워크플로우 엔진의 실행을 통하여 워크플로우의 수행을 정의, 생성, 관리하는 시스템이다. 본 연구에서는 기존의 워크플로우 시스템의 표준화 필요성을 인식하고 기존 워크플로우 시스템의 한계점인 제품들간의 상호운용성 문제와 동적으로 변화하는 비즈니스 환경에 적용하는 문제 및 프로세스 분석이나 관리 감독기능이 부족하다는 점을 분석하여 이에 대한 해결방안을 제공하기 위한 워크플로우 시스템의 설계 및 구현에 중점을 두었다. 빌드타임(Build-Time)에 정의된 프로세스의 정의를 런타임(Run-Time)에 동적으로 변경할 수 있도록 설계, 구현하였으며, 워크플로우 엔진과 관련 부분들의 기능을 WfMC(Workflow Management Coalition) 명세 및 API에 준하여 개발함으로써 상호운용이 가능하게 하였다.

Design and Implementation of Workflow Management System

Dong-Il Shin† · Dong-Kyoo Shin††

ABSTRACT

Workflow means the automation of a business process, in which documents, information or tasks are transferred among participants for business action according to a set of procedural rules, and workflow management system is a system which defines, creates and manages the execution of workflow running one or more workflow engines. In this research, necessity of standardizing current workflow systems is recognized, and problems such as interoperability of current systems, dynamic adaptation to changing business environment and lacking of assessment, management and auditing of business processes are analyzed so that the design and implementation of a workflow system is focused on to offer a solution to the problems. The system is designed and implemented to change dynamically on Run-Time the processes definition defined on Build-Time, and interoperability is enabled by developing workflow engine and related modules based on the WfMC specifications and APIs.

1. 서 론

급속한 기업 환경의 변화, 기업간 치열한 경쟁 속에서 기업들은 생존을 위해 과학적인 경영 기법을 도입하여 경영 환경의 최적화와 고객 만족의 극대화를 이루려는 노력을 계속해서 진행시켜 왔다. 이에 따라 80년대 초반 업계 및 학계에 큰 영향력을 미치면서 파급되어진 비즈니스 프로세스 리엔지니어링(BPR : Business Process Reengineering)이 등장하였고, 프로세스의 관점에서 데이터베이스 및 각종 응용업무의 정보자원들과 사용자들 사이에 발생하는 상호 작용과 협동작업을 묘사하는 비즈니스 프로세스의 행위 모델을 지원하는 정보기술이 필요하게 되었다. 워크플로우 관리

† 종신회원 : 세종대학교 컴퓨터공학과 전임강사
†† 종신회원 : 세종대학교 컴퓨터공학과 조교수
논문접수 : 2000년 3월 24일, 심사완료 : 2000년 4월 28일

시스템(WfMS : Workflow Management System)은 이러한 필요성에 의해 등장하게 되었으며, 학계나 업계, 컨설팅 영역에서 주목을 받고 있다. 워크플로우 관리 시스템은 업무 흐름을 관리하는 일종의 동적 시스템으로서, 비즈니스 프로세스 자동화 도구로서 정의되는 등, 다양한 개념으로 정의된다. 일반화된 개념으로서 워크플로우 관리시스템이란 기업 내외의 업무들과 관련된 사람들과 정보 및 기타 자원들의 흐름을 통합적으로 관리, 지원하는 자동화 도구들의 집합이라고 할 수 있다. 워크플로우 관리 시스템은 기업의 업무 흐름을 자동으로 처리함으로써 이에 따른 비용과 시간을 절감할 수 있는 솔루션으로 외국에서는 제조, 금융, 의료 등의 분야에서 폭 넓게 도입돼 기업의 생산성 향상 도구로서 자리를 잡았고, 국내에서도 제조, 금융 등의 분야에서 적극적인 도입하고 있다.

다음은 워크플로우 관리시스템의 표준화를 진행하고 있는 WfMC(Workflow Management Coalition)에서 선정한 관련용어들이다[1].

- 비즈니스 프로세스 : 일반적으로 기능적인 역할과 관계를 정의하는 조직구조의 맥락에서 업무 목표나 정책적인 목적을 총괄적으로 실행하는 일련의 연관된 업무절차.
- 워크플로우 : 전체적인 또는 부분적인 비즈니스 프로세스의 자동화를 의미하며, 이때 문서, 정보, 태스크가 한 사용자에서 다른 사용자로 일련의 업무절차 규칙에 의한 처리를 위해 전달된다.
- 워크플로우 관리시스템 : 하나 또는 그이상의 워크플로우 엔진을 실행하는 소프트웨어의 이용을 통하여 워크플로우의 수행을 정의, 생성, 관리하는 시스템을 의미하며, 여기서 워크플로우 엔진은 프로세스 정의를 해석하며 워크플로우 사용자와 상호작용하며 적절한 IT(Information Technology) 도구와 응용 업무들을 실행한다.

이상의 용어 정의에서 알 수 있듯이 워크플로우 관리시스템의 핵심은 워크플로우 엔진의 구축이다. 워크플로우 엔진을 이용함으로써 태스크들을 조직화하고, 스케줄링, 통제, 모니터링을 할 수 있다[2].

본 연구에서는 워크플로우 관리 시스템의 국제 표준을 제정하는 WfMC에서 규정한 표준문서에 기반 하여 워크플로우 관리시스템의 핵심기술인 워크플로우 엔진과 워크플로우 정의도구 및 워크플로우 클라이언트을 개발하였다. 개발에는 플랫폼에 독립적인 자바 언어를 사용하였다. 본 논문의 구성은 다음과 같다. 2장은 워크플로우 관리 시스템의 연구동향과 워크플로우 표준화의 필요성에 대해서 서술한다. 3장에서는 효율적인 워크플로우 관리를 위한 워크플로우 관리 시스템의 특성과 전체 시스템의 구조를 제안하고 주요 모듈별 기능 정의에 대하여 기술한다. 4장에서는 3장의 설계에 대한 구현에 관하여 기술한다. 마지막으로 5장에서는 결론과 워크플로우 관리 시스템의 향후 연구 방향에 대해서 기술한다.

2. 관련 연구

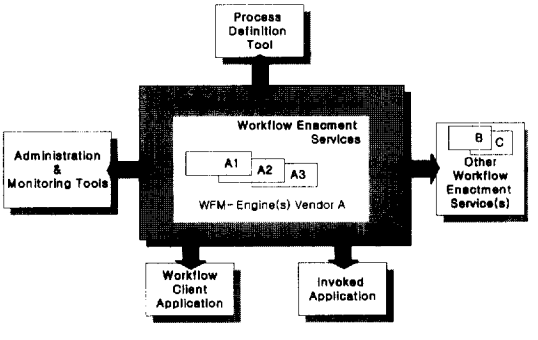
2.1 워크플로우 관리시스템의 개발과 표준화의 필요성

기존의 전통적인 수작업에 의한 비즈니스 프로세스의 진행에는 다음과 같은 문제점들이 내재한다[1]. 첫째, 종이문서 기반(Paper-based) 정보의 전달에 의존해야만 한다. 둘째, 업무가 거의 사람에 의존한다. 셋째, 책임 전가의 문제 발생의 가능성이 높다. 넷째, 업무의 추적(Tracking)에 많은 어려움이 있다. 다섯째, 프로세스 시간(Process Time)과 비용(Cost)을 효과적으로 측정할 수 없다. 그러나, 워크플로우 관리 시스템을 도입하여 자동화된 비즈니스 프로세스를 구현할 때에는 다음과 같은 장점들이 있다. 첫째, 업무흐름의 자동화 및 진행 상황 추적, 관리가 용이하다. 둘째, 정보분석 및 의사 결정의 신속성을 얻을 수 있다. 셋째, 종이문서 없는(Paper-free) 환경 구축이 가능하다. 넷째, 빠르고 원활한 의사소통이 가능하다. 다섯째, 신속한 업무 대처가 이루어진다. 여섯째, 폭 넓은 정보공유가 이루어진다. 따라서 기존 수작업에 의한 비즈니스 프로세스는 여러 가지 문제점이 내재되어 있고 워크플로우 관리 시스템의 도입으로 인한 장점들이 뚜렷하게 나타나므로 워크플로우 관리 시스템의 개발의 중요성이 점차적으로 증가하게 된다.

워크플로우는 비즈니스의 목적으로 업무에 참여하는 구성원들 사이에서 이루어지는 문서와 정보 또는 작업의 절차를 일종의 정의된 규칙의 집합에 따라 자동화하는 것이다. 따라서 워크플로우란 전체 또는 모든 분야에서의 비즈니스 프로세스의 자동화 또는 컴퓨터 처리를 의미한다고 할 수 있다. 워크플로우는 종종 비즈니스 프로세스 리엔지니어링과 관련지어 생각되기도 한다[3]. 비즈니스 프로세스 리엔지니어링이 프로세스

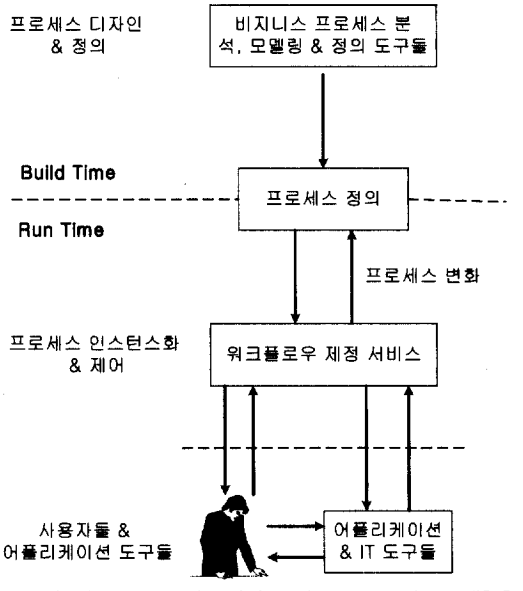
그 자체를 대상으로 하여 조직 구조와 업무 전반의 근본적인 개혁을 시도하는 것이라면, 워크플로우 관리 시스템은 변화된 프로세스가 정보 시스템 상에서 올바르게 수행되도록 하는 도구의 역할을 한다. 워크플로우 관리 시스템은 업무 흐름을 자동화하고 정보 및 문서의 전달을 전자화하며 일관성 있는 데이터 접근 및 제어를 통해 비지니스 프로세스를 개선, 통제, 관리하며 공동 작업을 지원하는 소프트웨어를 일컫는다[2]. 워크플로우 관리 시스템은 컴퓨터 소프트웨어의 실행을 통해 워크플로우를 정의하고 관리하고 실행하는 시스템으로서 작업과 그 작업의 여러 활동단계에 따르는 적절한 자원을 정의하고 작업 흐름의 관리에 의해 비즈니스 프로세스를 자동화하는데 이용된다.

이런 워크플로우 관리 시스템은 현재까지 서로 다른 제품의 연동에 구애받지 않고 사용할 수 있는 표준이 정의되지 않았기 때문에 시스템 자동화의 커다란 장벽으로 남아 있다. 현재 WfMC에서는 워크플로우 관리 시스템 간의 상호 운용성과 워크플로우 제품의 호환성을 해결하기 위해 워크플로우 제품을 위한 5가지 인터페이스 및 메타모델 표준화를 주도하고 있다. WfMC는 모든 워크플로우 제품들이 제공해야 할 다양한 기능들 중에서 상호 운용이 가능한 공통적인 기능들을 구분해내고 워크플로우 제품 구현을 위한 적절한 세부 규정을 발전시키기 위해 활동하고 있다. 이러한 공통된 규정을 통해 WfMC 권고안에 따라 구현된 동종의 워크플로우 제품들 사이에서는 상호 운용이 가능하다. 다음은 WfMC에서 제시하는 5가지 인터페이스이며 그 연관관계는 (그림 1)과 같다[1].



(그림 1) WfMC에서 제시하는 5가지 인터페이스[1]

- 프로세스 정의 교환 인터페이스(Process Definition Interchange Interface)
- 워크플로우 클라이언트 인터페이스(Workflow Client Application Interface)
- 호출된 응용 인터페이스(Invoked Application Interface)
- 워크플로우 상호운용 인터페이스(Workflow Interoperability Interface)
- 시스템 관리 감시 인터페이스(System Administration & Monitoring Interface)



(그림 2) 워크플로우 시스템의 구성 - WfMC 참조모델[1]

일반적인 워크플로우 관리시스템은 (그림 2)에서 보는 바와 같이 기능적으로 빌드타임(Build-Time)과 런타임(Run-Time) 모듈로 구분된다[1]. 비즈니스 프로세스를 분석하고 모델링 하여 정의를 하는 작업이 빌드타임 모듈에서 이루어지고, 프로세스를 진행하고 통제하며 동적으로 변경하는 작업이 런타임 모듈에서 수행된다. 빌드타임 모듈에서 분석, 모델링 되어 프로세스 정의도구(Process Definition Tool)를 통해 정의된 프로세스는 런타임 모듈에서 프로세스 인스턴스(Process Instance)로 만들어져 여러 개의 태스크들이 수행된다. 이 과정에서 많은 프로세스의 다양한 인스턴스들의 진행을 관리 감독하는 것이 워크플로우 엔진이다[3]. 태스크 수행을 담당하는 사용자들은 클라이언트 애플리케이션을 통해서 업무를 처리한다. 따라서, 워크플로우 관리 시스템을 사용함으로써 태스크들을 조직화하고, 스케줄링, 통제, 모니터링을 할 수 있다.

2.2 워크플로우 연구동향

워크플로우라는 개념은 오랫동안 존재해 왔지만, 워크플로우 관련 연구가 전세계적으로 활발히 진행되고 있는 것은 최근 몇 년간의 일이다. 최근의 연구동향은 워크플로우 정의와 설계, 태스크간의 종속성과 스케쥴링에 관한 연구, 워크플로우 관리시스템의 설계로 나눌 수 있다. 많은 논문들이 워크플로우 모델링, 워크플로우 정의와 워크플로우 설계에 관하여 발표되었다[4-6]. 또한 워크플로우 관리에 트랜잭션의 개념을 도입한 트랜잭션 기반의 워크플로우가 발표되었고 계속적인 연구가 이루어지고 있다[5, 7, 8]. 태스크간의 종속성에 대한 정의와 강화에 대한 연구는 트랜잭션의 구조와 행위를 정의하려는 노력에서 시작되었다[9, 10]. 조건적인 존재 종속성에 대한 정의가 진행되었고, 태스크 상태 전이 다이아그램이 태스크의 구조를 기술하기 위하여 이용되었다[9]. 다양한 데이터베이스 연산 관련 종속성이 또한 정의되었다[8, 11]. 워크플로우 관리시스템의 설계는 워크플로우 정의에 의존하기 때문에 이에 대한 연구는 다른 분야에 비해 늦게 시작하였다. 중앙 집중식 아키텍처인 FlowMark가 설계되었고[12], FlowMark는 워크플로우 관리 시스템의 분산 아키텍처로 확장되었다[13]. 이외에도 Object Flow 아키텍처[4]등의 아키텍처가 있으며 이외에 새로운 종류의 혁신적인 아키텍처는 아직까지는 발표되지 않은 상황이다. 워크플로우 신뢰성과 회복성에 관한 연구도 최근 진행되고 있으며, 특히 대규모 워크플로우 관리 시스템에서의 오류처리에 대한 연구가 진행되고 있다[3, 6].

워크플로우의 개념과 인터페이스를 표준화하려는 노력은 주로 WfMC에 의해 이루어지고 있다[1]. WfMC는 워크플로우의 5가지 인터페이스 및 메타모델의 표준화를 진행 중에 있으며, 이에 따른 많은 상용 제품들이 출시되고 있다. 분산 환경에서 동작하는 워크플로우 관리 시스템을 위하여 OMG(Object Management Group)와 WfMC가 협력하고 있으며, OMG의 분산 객체 관리 표준인 CORBA(Common Object Request Broker Architecture)[14]를 워크플로우 관리 시스템에 적용하려는 연구가 이루어지고 있다.

2.3 기존의 워크플로우 관리 시스템들

1990년대 초에 여러 회사들이 각종 워크플로우 시스템을 제작하여 공급하기 시작하였다. 이 중에서 대표적인 몇 가지 시스템에 대하여 소개하면 다음과 같다.

- 액션 워크플로우 시스템(Action Workflow System) : 액션 테크놀로지 사에서 공급하는 제품으로서 현재 마이크로소프트 SQL 서버용과 로터스 Notes용의 두 가지 버전이 있다. 이 시스템은 기본적으로 세 개의 부분으로 나뉜다. 액션 워크플로우 관리 시스템은 워크플로우 트랜잭션을 수집하고 조절하는 역할을 한다. 분석가(Analyst)는 워크플로우 프로세스를 디자인하기 위한 편집 도구이며, 애플리케이션 빌더(Application Builder)는 분석가를 통해서 만들어진 프로세스 정의를 실제 실행 가능한 프로세스로 만들어서 액션 워크플로우 관리 시스템으로 넘기는 역할을 한다. 이 외에도 각종 워크플로우 처리 상황을 프린트하기 위한 리포터(Reporter)도 제공된다.
- FlowMark[12] : IBM의 대표적인 워크플로우 제품이다. 여러자기 운영체제에서 동작되며 ODI사에서 제공하는 ObjectStore라는 객체지향 데이터베이스를 이용한다. FlowMark는 서버/빌드타임 클라이언트/런타임 클라이언트 및 프로그램 실행 클라이언트로 구성된다. 서버는 기본적으로 데이터베이스와 연결되어서 워크플로우 실행을 조절하는 역할을 한다. 빌드타임 클라이언트는 워크플로우 프로세스를 디자인하기 위한 도구이다. 런타임 클라이언트는 워크리스트에 대한 처리를 담당하는 도구이며, 프로그램 실행 클라이언트는 각종 API(Application Programming Interface)와 표준 인터페이스 모듈 기능을 제공한다.
- Workflo 비즈니스 시스템 : FileNet사에서 제공하는 제품으로서 오라클 데이터베이스를 이용한다. 워크플로우 데스트탑, Workshop, WorkFlo, FolderView 및 워크플로우 애플리케이션 라이브러리등의 여러 가지 부분으로 구성된다.
- ProcessIT : AT&T의 제품으로서 유닉스에서 동작되며 SQL 데이터베이스 기반으로 구성되었다. 워크플로우 트랜잭션에 기반한 시스템으로서 4가지 부분으로 나뉜다 : MapBuilder(프로세스를 정의하기 위한 윈도우 기반의 인터페이스 제공), Process Activity Manager(실질적인 워크플로우 엔진이다), WorkView(워크리스트 인터페이스를 제공한다), ProcessIT(시스템의 현 상태를 감시하고 문제점을 발견하는 도구이다).

3. 워크플로우 관리 시스템의 설계

본 연구의 목적은 워크플로우 엔진을 WfMC 정의문서 및 API(Application Programming Interface)에 준하여 개발함으로써 워크플로우 시스템의 구축의 기반 기술을 확보함을 목적으로 한다.

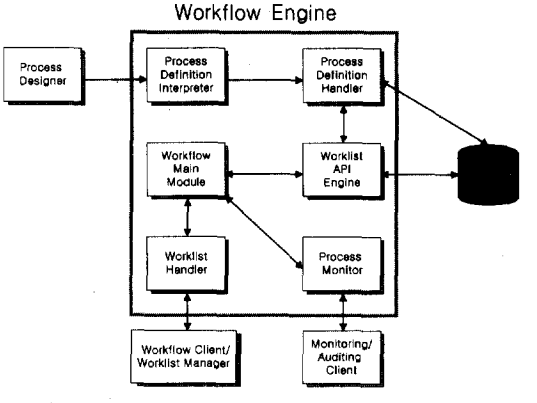
3.1 효율적 워크플로우 관리를 위한 워크플로우 관리 시스템

본 연구에서 개발된 워크플로우 관리 시스템은 빌드타임에 정의된 프로세스의 정의를 런타임에 동적으로 변경할 수 있도록 설계한 구조이다. 우선 업무 프로세스가 정의되어 프로세스 데이터베이스에 저장된 프로세스라면 언제라도 그 변경이 가능하도록 하였다. 다만, 프로세스 정의의 변경은 시스템 관리자만이 가능하도록 하여 일반 사용자는 사용할 수 없도록 제한하였다. 이에 대한 상세한 설계 및 구현 내용은 다음 장에서 설명된다. 또한 본 연구에서 개발된 워크플로우 관리 시스템은 WfMC에서 제시한 워크플로우 제품을 위한 5가지 인터페이스인 첫째, 프로세스 정의 교환(Process Definition Interchange). 둘째, 워크플로우 클라이언트 애플리케이션(Workflow Client Application). 셋째, 호출된 응용(Invoked Applications). 넷째, 워크플로우 상호운용(Workflow Interoperability). 다섯째, 시스템 관리 감시(System Administration & Monitoring)에 준하여 설계하였고 또한 메타모델 표준화 규정에 따라 개발되었다.

3.2 워크플로우 관리 시스템의 설계

WfMC에서 제시하는 지침에 따라 워크플로우 관리 시스템을 (그림 3)과 같이 설계하였다. 각 구성 모듈의 기능은 다음과 같다.

- 프로세스 디자이너(Process Designer) : 워크플로우 프로세스를 관리자가 비쥬얼(Visual)하게 정의하고 이를 WPDL(Workflow Process Definition Language)의 형태로 저장하여 비즈니스 프로세스를 정의하고 관련 규칙을 입력할 수 있는 비쥬얼(Visual) 도구
- 프로세스 정의 해석기(Process Definition Interpreter) : 프로세스 디자이너에서 생성된 WPDL을 해석하여 내부 데이터 구조에 적합한 형태로 변환
- 프로세스 정의 처리기(Process Definition Handler) : 프로세스 정의 데이터를 데이터베이스에 저장하고 인출하는 기능을 담당
- 워크플로우 메인 모듈(Workflow Main Module) : 정의된 프로세스들의 시작부터 종료까지의 모든 조절을 담당
- 워크플로우 API 엔진(Workflow API Engine) : 클라이언트 어플리케이션이 워크플로우 엔진의 기능을 이용할 수 있도록 WfMC 표준안에 준거하여 Java API를 제공[16].
- 워크리스트 처리기(Worklist Handler) : 워크플로우 엔진과 워크리스트 클라이언트 사이에서의 통신을 담당
- 프로세스 모니터(Process Monitor) : 모든 저장된/진행중인 프로세스의 Auditing 임무를 담당
- 워크플로우 클라이언트/워크리스트 관리자(Workflow Client/Worklist Manager) : ERP(Enterprise Resource Planning), EDMS(Electronic Document Management System) 및 기타 IT(Information Technology) 도구들이 워크플로우 엔진의 기능을 사용할 수 있는 API를 제공하여 기존의 IT 시스템과 워크플로우 엔진이 연동될 수 있는 통로 역할을 수행함. 워크리스트 관리자는 일반 사용자에게 워크플로우 엔진에 의해 처리되는 워크리스트를 전달하고 이의 처리여부를 워크플로우 엔진에게 알리는 역할을 수행함.



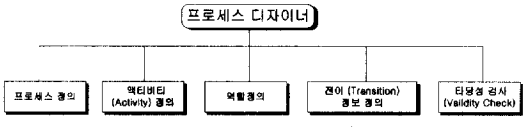
(그림 3) 워크플로우 관리 시스템 전체 설계도

위의 설계를 기반으로 워크플로우 엔진 및 관련부분 기능 정의를 수행하였으며, 그 결과 워크플로우 관리 시스템을 프로세스 디자이너, 워크플로 엔진, 워크플로우 클라이언트의 일부분인 워크리스트 관리자의 세 부

분으로 구성하였다.

3.2.1 프로세스 디자이너(Process Designer)

프로세스 정의도구인 프로세스 디자이너는 그래픽 환경에서 프로세스를 모델링하는 프로그램으로 사용자가 업무시스템 전체의 작업흐름을 설계하고 이를 데이터베이스에 저장할 수 있다. 이와 함께 역할 정의기(Role Definer)를 사용하여 업무시스템에 관여하고 있는 인력자원을 정의하고 관리하여 프로세스 모델링에 사용한다. 프로세스 디자이너는 프로세스 디자인에 대한 메타정보와 WPDL형식의 정보를 저장하며, 이 정보가 프로세스정의 해석기에 보내져 해석되어 데이터베이스에 저장된다. WPDL을 통하여 프로세스정의를 분산 환경하에서 파일단위로 쉽게 교환하거나 공유할 수 있다[17]. 프로세스 정의기는 (그림 4)에서 보는바와 같이 프로세스 정의, 액티비티(Activity)정의, 전이 정보(Transition Information) 정의, 역할 정의, 타당성 검사(Validity Check) 기능이 있다. 프로세스 정의기는 그 사용을 제한하여 일반 사용자가 아닌 관리자 및 특수 사용자만이 가능하도록 하였으며, 관리자가 특수 사용자의 범위를 일반 사용자 중에서 선정할 수 있도록 하였다.

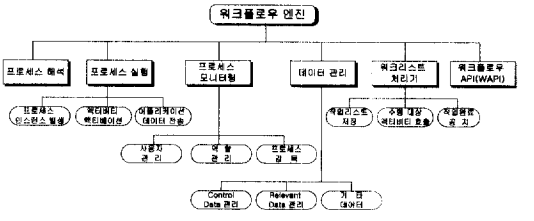


(그림 4) 프로세스 정의기 기능도

3.2.2 워크플로우 엔진

워크플로우 엔진은 데이터베이스에 저장된 프로세스 정의를 이용하여 업무를 프로세스 참여자에게 할당하고 전체 업무흐름을 감독 통제한다((그림 5) 참조). 수행 대상 태스크를 프로세스 진행 상황에 따라 선택하여 특정 업무담당자의 태스크 관리자(Task Manager)에 할당한다. 이 밖에 전체 프로세스관리, 통계 자료제공, 단일 프로세스 인스턴스에 대한 모니터링을 수행하기 위한 기능을 제공한다.

또한, 설계된 프로세스가 여러 가지 업무 환경의 변화에 따라 변경되어야 할 필요성은 항상 존재한다. 이에 따라서 프로세스 정의기를 통해서 설계된 프로세스 정의는 프로세스 입출력 인터페이스를 통해서 워크플로우 엔진에 전달되며, 엔진은 이를 워크플로우 데이타베이스에 저장한다[15]. 이 프로세스 정의 데이타베이스를 바탕으로 프로세스 인스턴스(Process Instance)가 발생할 때마다 프로세스 인스턴스 정보가 생성되어 데이타베이스에 저장된다. 런타임에 태스크들의 담당자, 작업기한, 내용 등의 정보가 동적으로 변경될 수가 있다. 이것은 프로세스 정의 템플릿으로 생성된 프로세스 인스턴스의 내용이 저장된 데이타베이스를 직접 접근하여 수정함으로써 가능하다. 런타임에 데이타베이스를 변경함으로써 워크플로우 엔진은 변경된 데이타베이스를 참조하여 프로세스를 진행시킨다.



(그림 5) 워크플로우 엔진 기능도

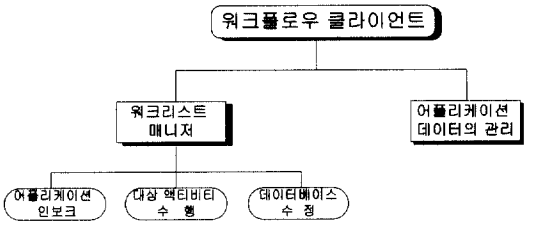
워크플로우 엔진은 다음과 같은 기능을 가지는 세부 모듈들로 나뉜다.

- 프로세스 해석 : 프로세스 정의 언어 해석 기능, 프로세스 정의 저장 및 인출 기능
- 프로세스 실행 : 프로세스 인스턴스 발생, 액티비티 실행, 어플리케이션 데이터 전송
- 프로세스 모니터링 : 사용자 관리 기능, 역할 관리 기능, 프로세스 감독 기능
- 데이터 관리 : 제어 데이타(Control Data) 관리, 관련 데이타(Relevant Data) 관리, 기타 데이터 관리
- 워크리스트 핸들러 : 워크 저장, 수행대상 프로세스 호출, 작업완료의 공지
- 워크플로우 API(WAPI) : 연결 기능, 프로세스 조절 기능, 액티비티 조절 기능, 프로세스 상태 기능, 액티비티 상태 기능, 워크리스트 관리 기능, 관리자 기능

3.2.3 워크플로우 클라이언트

워크플로우 클라이언트는 사용자들을 위해 구현되었으며 담당자별로 할당된 워크리스트를 관리한다((그림 6) 참조). 클라이언트 모듈은 할당된 태스크들을 관리

하는 워크리스트 처리기(Worklist Handler)로부터의 정보를 이용하여 사용자가 해당되는 액티비티를 수행하는 워크리스트 관리자가 있고 워크플로우 관리자 자체의 데이터를 관리하는 기능이 있다.



(그림 6) 워크플로우 클라이언트 기능도

- 액티비티의 수행 : 어플리케이션 인보크, 대상 액티비티 수행, 데이터베이스의 수정
- 어플리케이션 데이터의 관리

4. 워크플로우 관리 시스템의 구현

4.1 프로세스 디자이너(Process Designer)

프로세스 디자이너는 워크플로우 엔진에서 처리해야할 작업을 정의하는 도구로서 사용자가 비주얼 도구를 사용함으로서 사용자가 작성하기 원하는 업무프로세스를 끌어당기기(Drag and Drop)방식으로 정의하는 프로세스 디자이너를 구현하였다. 사용자에 의해 정의된 정보는 각종 그래픽 객체에 대한 일련화(Serialization)를 위해서 프로세스 디자이너의 자체 파일 포맷으로 저장된 후에 필요한 데이터의 파싱을 통해서 WPDL로 변환 및 저장되고, 이 WPDL 정보가 워크플로우 엔진에 보내져 최종적으로 데이터베이스에 저장된다. 프로세스정의 도구의 모습과 주요부분은 (그림 7)과 같다.

- 사용자 인증 부분

프로세스 디자이너는 편집 시에 이용되는 참여자 리스트를 워크플로우 데이터베이스에 저장되어 있는 사용자(Participant) 테이블에서 가져오기 때문에 기본적으로 인증된 사용자만이 서버에 접속한 후 사용할 수 있다.

- 편집 창 및 편집 도구

사용자는 편집 창에서 편집도구 아이콘을 끌어당기기(Drag and Drop) 방식으로 프로세스를 정의한다. 여기서 사용되는 노드들은 다음과 같이 구분된다.

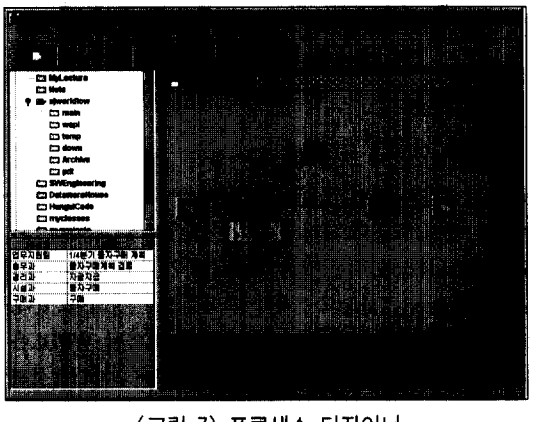
- 시작정보 - 시작정보에는 처리되어야할 문서나 어플리케이션 정보를 지정하고 시작과 종료 기한의 설정, 수신자 지정 등의 세부 기능이 정의된다.
- 태스크 정보 - 문서나 어플리케이션 정보, 수신자 지정, 시작조건, 처리결과 통보기능 정의
- 종료정보 - 종료 조건, 종료 결과 통보 기능이 정의
- 서브플로우(Subflow) 정보 - 서브플로우는 하나의 태스크 및 업무 프로세스를 정의함으로써 프로세스 내의 프로세스를 정의한다. 따라서 새로운 업무 프로세스 정의를 첨부하는 기능이 포함되어 있다.
- 전이(Transition) 정보 - 각 액티비티의 진행 조건을 정한다. 진행조건은 And-Split, And-Join, OR-Split, OR-Join으로 나뉜다.

- 워크플로우 엔진과 연결 설정 해제

워크플로우 엔진과의 연결을 설정하거나 해제한다.

- 서버로 WPDL 전송

WPDL 화일을 선택하여 서버로 전송한다.

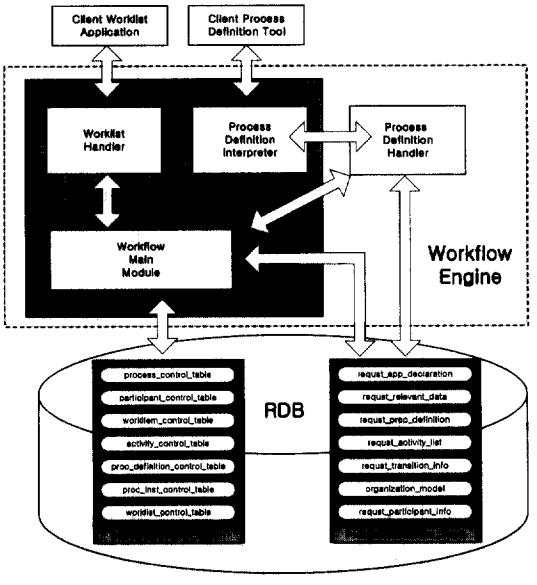


(그림 7) 프로세스 디자이너

4.2 워크플로우 엔진

워크플로우 엔진에서는 프로세스 디자이너로부터 전송된 WPDL 문서를 인터프리터에서 해석하고 프로세스정의 처리기에서 데이터베이스에 필요한 데이터를 저장한다. 워크플로우 엔진에서는 저장된 프로세스정의의 시작조건을 검사하여 프로세스의 인스턴스인 제어(Control) 테이블에 수행정보를 저장하고 프로세스의

태스크들을 진행시킨다. 이렇게 수행되어진 프로세스들은 각각의 사용자에게 부여된 작업들을 요청에 따라 전송해주며 각 사용자가 처리한 결과를 되돌려 받아 태스크의 진행을 처리한다. 워크플로우 엔진은 WfMC 명세서인 WAPI를 기준으로 구성되었으며 상세한 구성은 (그림 8)과 같다.



(그림 8) 워크플로우 엔진 구조

데이터베이스에 저장되는 테이블은 그림에서 볼 수 있듯이 런타임시 메인 모듈에서 프로세스, 액티비티, 워크리스트 생성 등을 위해서 사용하는 System_Control_table과 프로세스에 대한 원본 데이터를 가지고 있는 Process_Definition_Table의 2개의 테이블로 구성되어 있다. 클라이언트의 프로세스정의 도구에서 정의된 WPDL은 워크플로우 엔진의 프로세스정의 해석기에 의해 해석되고 프로세스정의 처리기는 이미 구성된 테이블에 필요한 데이터를 저장하고 메인 모듈에게 통보한다. 메인 모듈에서는 시작조건을 검사하여 프로세스 인스턴스를 위한 컨트롤 테이블에 데이터를 저장한다. 만들어진 인스턴스는 System_Control_table에 기록이 되며, 이 프로세스 인스턴스를 통해 각 액티비티 인스턴스가 만들어진다. 현재 실행되어져야 하는 인스턴스는 Worklist_Control_Table에 저장이 되며 이는 사용자가 워크리스트 관리자를 이용하여 가져온다. 워크리스트 처리기를 통해 사용자는 먼저 자신의 ID와 암호를 통해 엔진에 접속한다. 그러면 엔진은 그 사용자에게 워크리스트 핸들(Worklist Handle)을 부여한다. 이 핸들은 엔진의 제어시에 반드시 필요하며, 이것을 통해 엔진에서 제공하는 모든 기능에 접근하게 되는 것이다. 따라서 엔진은 각각의 작업에 대한 핸들을 부여하게 되며, 이를 통해 접근을 할 수 있도록 허용한다.

엔진의 모든 작업은 WAPI 표준에 맞게 실행된다. 우선 Java로 구현되어진 WfMC WAPI 명세가 없는 관계로 현재 나와 있는 명세를 참고하여 패키지를 구성하였다. 워크리스트 처리기는 클라이언트의 요청을 받아서 메인 모듈로 전달하게 된다. 메인 모듈의 메소드는 워크리스트 처리기에서 들어온 요청에 따라 여러 가지 작업들을 처리하게 된다. 주요 엔진 메소드와 그 설명은 <표 1>과 같다.

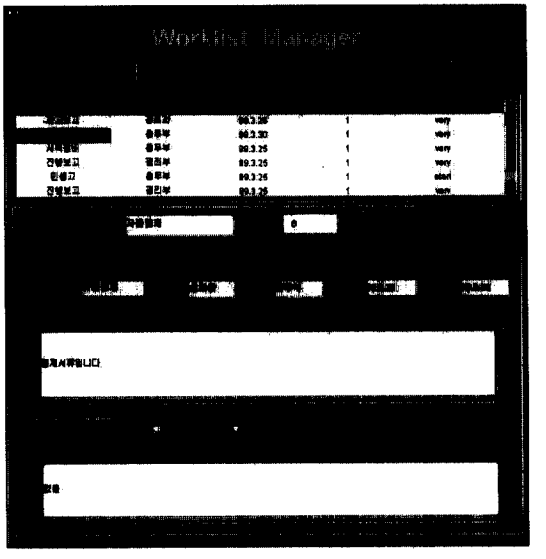
〈표 1〉 엔진 메소드

Class	Method()
workflow_engine	protected synchronized void addClient(Session_handle handle)
	public Workflow_Engine()
	protected boolean close_proc_def(String sh, String qh)
	protected boolean close_worklist(String s_handle, String query_handle, Session_handle handle)
	protected boolean create_process_instance(String s_handle, String def_id, String inst_name,Session_handle handle)
	protected boolean disconnect_sessionID(String sh)
	protected boolean get_sessionID(String name, String pswd, String engine, String scope, Session_handle handle)
	protected boolean get_workitem(String s_handle, String pproc_inst_id, String item_id, Session_handle handle)
	protected void getMessage(String s, Session_handle handle)
	public boolean init_db()
	protected boolean open_procdeflist(String sh, WMTFilter filter, boolean count_flag, Session_handle handle)
	protected boolean open_worklist(String sh, WMTFilter filter, boolean count_flag, Session_handle handle)
	protected synchronized void removeClient(Session_handle handle)
	public void run()
	public synchronized void stop(Session_handle handle)
	protected boolean terminiate_proc_inst(String sh, String p_id)
Session_handle	public Session_handle(Workflow_Engine engine, Socket socket)
	boolean connect()
	public void disconnect()
	public String host()
	public synchronized boolean put(String str)
	public void run()

대부분의 워크플로우 시스템들이 한번 정의된 프로세스를 따라서 업무를 완료하도록 하고 있는데, 실제 비즈니스 환경은 동적 변화가 자주 발생하기 때문에 본 연구에서 구현한 워크플로우 시스템에서는 관리자 및 특수 사용자가 직접 워크플로우 클라이언트를 이용하여 이를 조절을 할 수 있는 기능을 구현하였다. 관리자는 워크플로우 클라이언트를 통해서 현재 실행중인 프로세스를 중지시키고, 이후에 프로세스 정의기를 통해서 변경된 프로세스 정의는 프로세스 입출력 인터페이스를 통해서 워크플로우 엔진에 전달된다. 이 프로세스 정의 데이터베이스를 바탕으로 프로세스 인스턴스(Process Instance)가 발생할 때마다 프로세스 인스턴스 정보가 생성되어 데이터베이스에 저장된다. 런타임에 태스크들의 담당자, 작업기한, 내용 등의 정보가 동적으로 변경될 수가 있다. 또한 현재 진행중인 프로세스에 대한 모든 정보 데이터베이스를 변경함으로써 워크플로우 엔진은 변경된 데이터베이스를 참조하여 프로세스를 진행시킨다.

4.3 워크플로우 클라이언트

워크플로우 클라이언트는 일반 사용자들을 위해 구현되어야 하기 때문에 사용자들이 쉽게 업무를 수행할 수 있도록 (그림 9)와 같이 구현하였다. 사용자는 자신에게 할당된 워크리스트를 클라이언트 어플리케이션에서 처리하고 수행된 업무들은 워크플로우 엔진에 전달하고, 클라이언트 어플리케이션은 할당된 업무들을 관리할 수 있도록 구현하였다. 프로세스 정의기에 의해 정의된 프로세스의 액티비티들이 클라이언트 어플리케이션에 의해 처리, 수정된다. 수정된 액티비티들은 워크플로우 엔진에 전달되고 워크플로우 엔진은 데이터베이스에 접근하여 현재 동작중인 프로세스의 중단 및 수정을 가능하게 구현하였다. 구현된 워크리스트의 메인 화면은 (그림 9)와 같다.



(그림 9) 워크리스트 클라이언트

5. 결 론

본 연구에서는 기존의 워크플로우 시스템의 표준화 필요성을 인식하고 기존 워크플로우 시스템의 한계점인 제품들간의 상호운용성 문제와 동적으로 변화하는 비즈니스 환경에 적용하는 문제 및 프로세스 분석이나 관리 감독기능이 부족하다는 점을 분석하여 이에 대한 해결방안을 제공하기 위한 워크플로우 시스템의 설계 및 구현에 중점을 두었다.

본 연구의 수행을 통해서 개발된 워크플로우 관리 시스템의 특징은 다음과 같다. 첫째, 본 시스템은 빌드타임에 정의된 프로세스의 정의를 런타임에 동적으로 변경할 수 있도록 구현하였다. 이것은 프로세스 정의 템플릿으로 생성된 프로세스 인스턴스의 내용이 저장된 데이터베이스에 접근하여 수정함으로써 가능하도록 구현하였다. 런타임에 데이터베이스를 변경함으로써 워크플로우 엔진은 변경된 데이터베이스를 참조하여 프로세스를 진행시킨다. 둘째, 워크플로우 엔진과 관련 부분들의 기능을 WfMC 명세 및 API에 준하여 개발함으로써 WfMC 명세 및 API에 준하여 개발된 제품들간의 상호운용이 가능하게 되었다. 셋째, 플랫폼 독립적인 언어인 Java를 사용하여 구현하므로 분산 환경에서의 작업 수행을 가능하도록 구현하였다.

본 연구에서는 향후에 오류에 대한 대응을 위하여 로그 저장 및 처리 모듈을 추가 연구 및 구현하고, 또한 감사/모니터링 기능을 추가 연구 및 구현하며, 좀 더 사용하기 쉬운 클라이언트 인터페이스의 구현을 위한 지능형 에이전트의 탑재를 목적으로 연구를 진행할 계획이다.

참 고 문 헌

[1] WfMC, "WfMC(Workflow Management Coalition) Standard Documents," Technical report, Workflow Management Coalition, November 1998.

[2] C.Mohan, "Tutorial : State of the Art in Workflow Management System Research and Products," IBM Almaden Research Center, Workflow Systems and Interoperability, August, 1997.

[3] D. Georgakopoulos, M. Hornick, A. Sheth, "An Overview of Workflow Management : From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Database, April 1995.

[4] M. Hsu and C. Kleissner, "ObjectFlow : Towards a process management infrastructure. Technical report," Digital Equipment Corporation, 1995.

[5] A. Sheth and M. Rusinkiewicz, "On transactional workflows," IEEE Data Engineering Bulletin, Vol. 16, No.2 pp.1-4, June 1993.

[6] N. Krishnakumar and A. Sheth, "Managing heterogeneous multi-system tasks to support enterprise-wide operations" Distributed and Parallel Databases, Vol.3, No.2 pp.155-186, April 1995.

[7] Y. Breitbart, A. Deacon, H. Schek, and A. Sheth, "Merging application-centric and data centric approaches to support transaction-oriented multi-system workflows," SIGMOD Record, Vol.22, No.3, pp.23-30, September 1993.

[8] D. Georgakopoulos and M.F. Hornick, "A framework for enforceable specification of extended transaction models and transactional workflows," International Journal of Intelligent and Cooperative Information Systems, Vol.3, No.3 pp.599-617, 1994.

[9] J. Klein, "Advanced rule driven transaction management," In Proc. of the IEEE COMPCON, pp. 562-567, San Francisco, CA, IEEE Computer Society, 1991.

[10] U. Dayal, M. Hsu, and R. Ladin,. "A transactional model for long-running activities," In Proc. of the 17th International Conference on Very Large Data Bases, pp.113-122, Barcelona, Spain, September 1991.

[11] M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth. "Using flexible transactions to support multi-system telecommunication applications," In Proc. of the 18th Int'l Conference on Very Large Data Bases, pp.65-76, August 1992.

[12] F. Leymann and W. Altenhuber, "Managing business process as an information resource," IBM Systems Journal, Vol.33, No.2, pp.326-348, 1994.

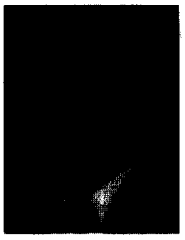
[13] G. Alonso, D. Agrawal, A, Abbadi, C. Mohan, M. Kamath, and R. Guenthoer, "Exotica/FMQM : A persistent message-based architecture for distributed workflow management," In Proc. of the IFIP Working Conference on Information Systems Development for Decentralized Organizations, pp.1-18, Trondheim, Norway, August 1995.

[14] OMG, "The common object request broker : Architecture and specification. revision 2.0," Technical report, Object Management Group, July 1995.

[15] 김동수, 배준수, 서영호, 허원창, 김영호, 강석호, "효율적 워크플로우 관리를 위한 시스템 특성 및 설계", 대한산업공학회/한국경영과학회, 1998.

[16] 강영식, 최인준, "웹, 자바, 코바를 기반으로 한 워크플로우 관리 시스템", 대한산업공학회/한국경영과학회, 1998.

[17] John A. Miller, Amit P. Sheth, Krys J. Kochut and Devanand Palaniswami, "The Future of Web-Based Workflows," Proc. of the Int'l Workshop on Research Directions in Process Technology, July 1997.



신 동 일

e-mail : dshin@sejong.ac.kr

1988년 연세대학교 전산과학과 (이학사)

1993년 M.S. in Computer Science, Washington State University

1997년 Ph.D in Computer Science, University of North Texas

1997년~1998년 시스템공학연구소 선임연구원

1998년~현재 세종대학교 컴퓨터공학과 전임강사

관심분야 : 이동통신, WAP, 멀티미디어 데이터베이스, CSCW, 지능형 에이전트

신 동 규

e-mail : shindk@sejong.ac.kr

1986년 서울대학교 계산통계학과 (이학사)

1992년 Illinois Institute of Technology 전산학과 (공학석사)

1997년 Texas A&M University 전산학과(공학박사)

1986년~1991년 한국국방연구원 연구원

1997년~1998년 현대전자 멀티미디어연구소 책임연구원

1998년~현재 세종대학교 컴퓨터공학과 조교수

관심분야 : 웹기반 멀티미디어, WAP, 멀티미디어 DB, 영상압축, 웨이브릿 부호화