2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

B2B 서비스를 위한 자바 기반의 워크플로우 관리 시스템의 Run-Time 설계 및 구현

이동훈*, 민덕기*, 김중배+, 김성훈+, 장철수+
*건국대학교 컴퓨터·정보통신공학과 +한국전자통신연구원
*{dhlee,dkmin}@cse.konkuk.ac.kr +{jjkim,saint,jangcs}etri.re.kr

Design and Implementation of a Java-Based WorkFlow Management System for Business-to-Business Service

Donghoon Lee*, Dugki Min*, Jungbae Kim+, Sunghun Kim+, CheulSoo Jang+
*Dept. of Computer Science and Engineering, Konkuk University
+Electronics and Telecommunications Research Institute

요 약

본 연구에서는 웹과 분산된 환경에서 신뢰할 수 있고 확장성 있는 워크플로우 관리 시스템의 구조와 워크플로우 관리 시스템의 핵심인 워크플로우 엔진의 구조를 설계하고 제안한 구조에 맞는 프로토타입 엔진을 구현하였다. 구현된 엔진은 다양한 Transition 타입을 지원하므로 실제 B2B환경에서 발생하는 다양한 형태의 비즈니스 타입을 지원한다. 또한 엔진 내의 인스턴트를 관리하는 인스턴트 관리자에 인스턴트들 사이의 처리 순서를 결정하는 인스턴트 스케줄링 기능이 들어있다. 이 인스턴트 스케줄링 기능은 인스턴트들을 사이에 우선순위를 부여한다든지 인스턴트의 실시간 보장을 위한 여러 관련 처리가 가능케 하여준다. 이외에 Invoked Application으로 웹 애플리케이션을 사용할 경우 Relevant Data를 어떻게 전달할 것인지, Notification 서비스, Monitoring 서비스 기능들을 고려하여 설계 및 구현하였다.

1. 서론

인터넷과 정보 통신 기술의 발전은 기업의 경영 환경을 급속하게 변화시키고 있다. 이런 변화하는 환경에서 기업은 경쟁력을 유지하기 위해서 ERP 시스템을 도입하였다. 이러한 ERP 시스템은 기업 환경이 급속하게 변화함에 따라 ERP 시스템의 변화를 요구하고 있으며 그 변화의 속도가 빨라지고 있다. 이런 전사적인 시스템은 업무의 흐름이 한번 정해지면 흐름을 변경하는 것이 어려웠고, 또한 복잡한 업무의 흐름을 구현하기가 어려웠다. 뿐만 아니라, 해당업무의 흐름을 분석하고 그의 개선 등을 파악하는데 많은 어려움이 있었다. 게다가 기존 기업들은 자신의 정보 시스템들을 보유하고 있었고 이런 정보 시스템들의 상호 호환되지 않는다는 문제가 있었다. 설령 비슷한 정보 시스템이라도 기업내부에서 사용되는 메타 정보들이 달라 통합하기가 쉽지 않았다. 이러한 요구를 반영하여 재 사용성이 가능하고 이질적인 정보를 가지는 정보 시스템들간의 상호 호환이 가능한 XML[6] 기반으로 한 분산 컴포넌트 방식[7,8]의 프로그램과, 정적인 업무의 흐름이 아닌 동적으로 업무의 흐름을 변경할 수 있으며 장시간에 걸쳐서 실행되는 프로세스의 지원과 비동기적인 실행을 보장하는 워크플로우 시스템[1,2,5]이 필요하게 되었다.

본 연구에서는 웹과 분산된 환경에서 신뢰할 수 있고 확장성 있는 워크플로우 관리 시스템의 구조와 워크플로우 관리 시스템의 핵심인 워크플로우 엔진의 구조를 설계하고 제안한 구조에 맞는 프로토타입 엔진을 구현하였다. 구현된 엔진은 다양한 Transition 타입을 지원하므로 실제 B2B환경에서 발생하는 다양한 형태의 비즈니스 타입을 지원한다. 또한 엔진 내의 인스턴트를 관리하는 인스턴트 관리자에 인스턴트들 사이의 처리 순서를 결정하는 인스턴트 스케줄링 기능이 들어있다. 이 인스턴트 스케줄링 기능은 인스턴트들을 사이에 우선순위를 부여한다든지 인스턴트의 실시간 보장을 위한 여러 관련 처리가 가능케 하여준다. 이외에 Invoked Application으로 웹 애플리케이션을 사용할 경우 Relevant Data를 어떻게 전달할 것인지, Notification 서비스, Monitoring 서비스 기능들을 고려하여 설계 및 구현하였다.

2장에서는 워크플로우 관시 시스템의 구조와 동작환경을 설명하였다. 이 장에서는 워크 플로우 관리 시스템 구조도에 대해 더 설명을 하도록 하겠다. 3장에서는 본 연구의 핵심 부분의 하나인 워크플로우 관리 시스템의 Run-Time 구조에 대하여 다루었다.

2. 워크플로우 관리 시스템

워크프로우 관리 시스템[3,4]은 핵심 컴포넌트인 워크플로우 엔진을 포함하는 워크플로우 인엑먼트 서비스를 중심으로 구성되어 있다. 워크플로우 엔진은 워크플로우 프로세스의 생성과 소멸, 진행에 대한 컨트롤, 여러 프로세스의 진행 스케줄 관리, 관리 시스템의 다른 부분들과 필요한 데이터 유지 및 전달 등의 핵심 기능을 수행한다. 이외에 워크플로우 관리 시스템은 새로운 프로세스 정의를 생성하는 프로세스 정의작성 툴, 워크플로우 진행 시 워크플로우 참여자와 상호작용을 하기 위한 부분, 워크플로우 프로세스 내의 엑티비티를 수행시 호출되어 실제 작업을 수행하는 어플리케이션 부분 그리고 엔진 관리자를 위한 관리 툴 등으로 구성되어있다.

본 논문은 워크플로우 인엑먼트 서비스를 중점적으로 설계하고 구현 하는데 초점을 맞추었다. 워크플로우 인액트먼트 서비스는 프로세스 정의를 해석하고 프로세스 인스턴스, 액티비티 시퀀싱, 사용자 Worklist에 work 아이템을 추가와 애플리케이션 툴의 호출을 제어하는 서비스이다. 이것은 하나 이상의 워크플로우 관리 엔진을 통해서 행해질 수 있고 다양한 프로세스의 개별적인 인스턴스의 실행을 관리한다. 워크플로우 인액트먼트 서비스는 워크플로우 엔진을 통해 집중화하거나 분산된 내부 컨트롤 데이터를 관리 할 수 있다. 워크플로우 엔진은 또한 특별한 액티비티 실행에 필요한 애플리케이션

을 활성화하기 위해 애플리케이션 툴 호출 능력을 포함하고 있다.

본 연구에서는 워크플로우 클라이언트로 웹 어플리케이션을 사용하였다. 최근에 비즈니스 환경은 전용클라이언트보다 웹환경이 일반화되어 가고 있기 때문이다. 워크플로우 엔진이 동작하기 위해서는 제반환경들이 모두 갖춰져야 한다. 여기서 제반환경이란 워크플로우 엔진 서비스를 사용하는 워크플로우 사용자 클라이언트와 실제 액티비티를 수행하기 위한 호출된 어플리케이션, 그리고 위 두 가지를 실행시킬 수 있는 웹 어플리케이션 서버 모두를 지칭한다. 사용자 클라이언트와 호출된 어플리케이션은 본 연구에서 웹 어플리케이션으로 설정되어 있기 때문에 기본적으로 웹 어플리케이션을 동작시킬 수 있는 환경이 필요하다. 웹 어플리케이션 서버는 Java를 사용할 수 있는 환경이면 어떠한 것이든 가능하다. Workflow Client는 워크플로우 사용자가 워크플로우를 제어하고 워크리스트를 보기 위해서 사용하는 어플리케이션이다. 여기에서 워크플로우 엔진의 원격 인터페이스를 사용해서 워크플로우 서비스를 이용한다. 워크플로우 클라이언트는 웹 어플리케이션이 될 수도 있고 전용 클라이언트가 될 수도 있다.

3. 워크플로우 관리 시스템 구조

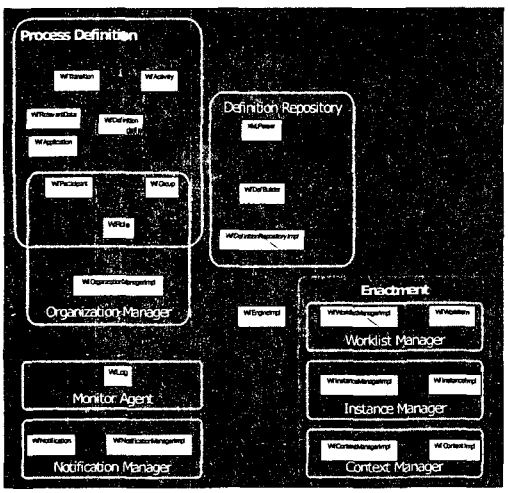

그림 1 워크플로우 엔진의 개략적 구조

그림 1은 본 논문에서 제시하는 워크플로우 엔진의 클래스 간의 관계만을 간단히 표현한 것이다. 이 워크플로우 엔진은 크게 6부분으로 나뉘어 진다. 여기서 Process Definition 객체그룹은 Build-time 시 XML로 표현된 프로세스 정의를 객체화 시킨것이고 Enactment 객체그룹은 Run-time시에 Process Instance를 생성하고 관리하는 엔진의 핵심에 해당한다. 나머지 그룹 지워진 객체들은 엔진과 연관된 부수적인 객체들이라고 볼 수 있다.

▶ Enactment Service

워크플로우 엔진의 핵심으로 Process Instance를 생성하고 진행하며 실행을 담당하는 역할을 수행한다. 인스턴트의 생성과 스캐줄 및 관리를 담당하는 Instance Manager, 인스턴트의 실행중인 Activity들의 Context를 생성하고 관리하는 Context Manager, Interactive Activity의 경우 담당자에게 처리를 요청하는 Workitem을 생성하고 관리하는 Worklist Manager 등이 동작하면서 다른 부분을 참조하며 워크플로우를 진행한다.

▶ Definition Repository

워크플로우 프로세스 정의 XML을 읽어서 Definition 객체를 생성하고 이를 관리하는 부분이다. 보통 워크플로우 엔진이 처음에 동작을 시작할 때 Definition객체를 구성하고 그 다음부터는 Definition객체를 참조하는 기능을 제공하게 된다.

▶ Process Definition

워크플로우 프로세스 정의 XML읽어서 생성되는 객체들이다. 앞 장에서 설명하였듯이 내부에 워크플로우 프로세스를 구성하는 데에 필요한 여러 다른 객체들을 포함하고 있다.

▶ Organization Manager

사용자, 그룹, 롤 관리를 제공한다. 사용자가 롤에 소속되어 있는지 검사하는 기능, 롤에 속해있는 사람들의 목록을 보는 기능 등을 제공한다. 이 부분은 현재는 워크플로우 관리 시스템의 한 부분으로 구성되어 있지만 원래는 Security Service의 한 부분으로 소속되고 워크플로우 관리 시스템에서 필요시에 사용하는 형태로 되어야 한다.

▶ Monitor Agent

워크플로우 엔진의 상태를 모으는 부분이다. 워크플로우 엔진에서 등록되어 있는 모든 정보의 현재상태를 Monitor 쪽으로 보고한다. 현재 Monitor에서는 보고된 상태를 화면에 현재 시간과 날짜와 같이 출력하고 있다. 이 부분은 나중에 Monitoring Service로 확장될 것이다.

▶ Notification Manager

사용자에게 보내지는 메시지를 관리하는 부분이다. 메시지는 워크플로우 엔진에서 사용자에게 보내기도 하고 다른 사용자가 보내기도 한다. 이 부분도 독립적인 Messaging Service로 확장시킬 수 있다. 본 연구에서는 비즈니스 프로세스의 종료 등 특별한 이벤트의 발생 시 관련 사용자에게 메시지를 보낼 때 주로 이용된다.

이상의 6개 부분에 속하는 객체들의 하나 하나를 자세히 살펴보도록 한다. 살펴보면 다음과 같다. 첫째로, WfEngine 객체는 워크플로우 엔진의 시작 객체이다. 워크플로우 엔진의 모든 기능은 이 객체를 통해서 사용할 수 있다.

3.1. Definition Repository 관련 객체

그림 2은 WfDefinitionRepository를 기준으로 그린 클래스구조

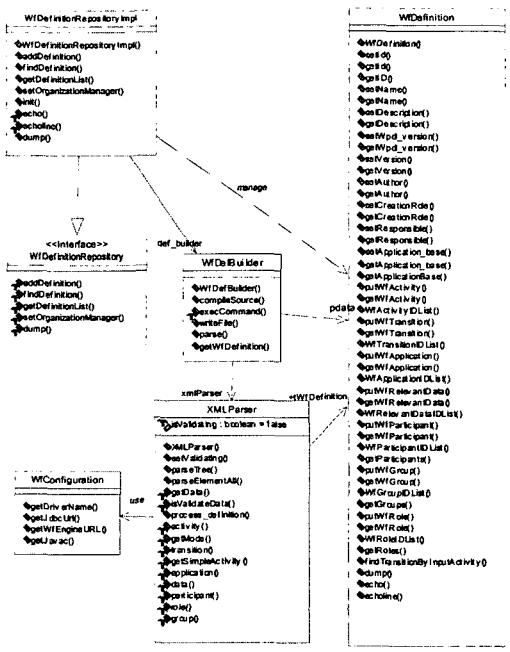

그림 2 DefinitionRepository 객체

도이다. DefinitionRepository는 워크플로우 엔진이 시작할 때 워크플로우 정의 XML을 읽어서 WfDefinition을 생성하고 이것을 관리하는 기능을 가지고 있다. 그림 2에서 XMLParser와 WfDefBuilder는 WfDefinition을 생성하기 위해서 워크플로우 정의 XML을 읽어서 분석하는 클래스이다. 이때 WfConfiguration의 getJavac()를 사용한다. 이것은 현재 시스템에서 Java Compiler의 위치를 돌려준다. WfDefinitionInfo는 워크플로우 프로세스 정의에 대한 정보를 가지고 있다. 사용자가 워크플로우 프로세스 정의를 얻을 때 객체 그 자체를 얻을 수는 없다. 그래서 간접적인 메타정보를 얻기 위해서 사용한다.

3.2 Instance Manager 관련 객체

그림 3은 WfInstanceManagerIml를 중심으로 해서 작성한 클래스구조도이다. WfInstanceManagerImpl은 WfInstnaceImpl을 관리하는 기능과 WfInstanceImpl의 수행을 스케쥴링해서 상태관리를 하는 기능을 가진 워크플로우 엔진의 핵심 객체이다. WfInstance는 워크플로우 프로세스 인스턴스를 나타낸다. 관련있는 context, definition 등과 연결되고 실제로 스케쥴 되었을 때 처리하는 코드를 가지고 있다. 또한 워크플로우 데이터를 가지고 있다. WfContext는 워크플로우 인스턴스와 Activity에 접근하기 위한 객체이다. Activity의 상태를 바꿔서 다음 단계로 진행하는 등의 행동을 취하게 한다. 워크플로우 컨텍스트는 WfContextManager에 의해 관리되며 워크플로우의 진행에 복제되거나 통합되기도 한다. 하나의 WfInstance에는 여러 개의 WfContext가 연결될 수 있다.

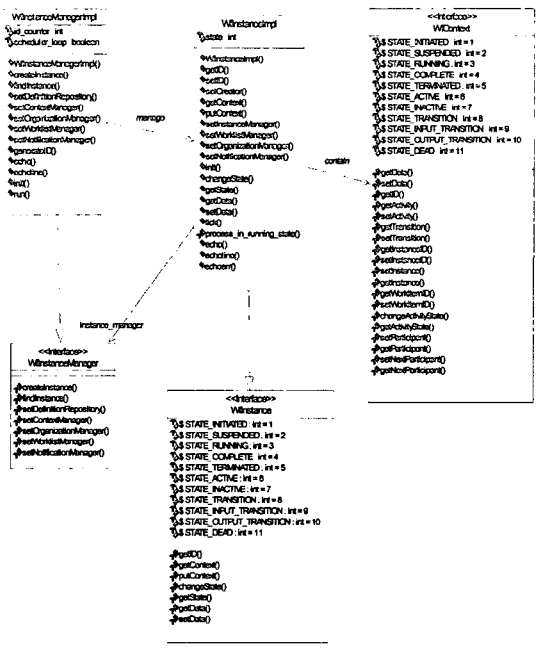

그림 3 InstanceManager 관련 객체

3.3 Worklist Manager 관련 객체

그림 4는 WfWorklistManager를 기준으로 해서 작성한 class diagram이다. WfWorklistManager는 다른 관리 객체에 비하면 상대적으로 간단한 기능을 가지고 있다. 사용자별로 WfWorkitem의 목록을 관리한다. 그래서 사용자가 자신의 Worklist를 요구하면 그 사용자의 Workitem을 뽑아서 되돌려준다.

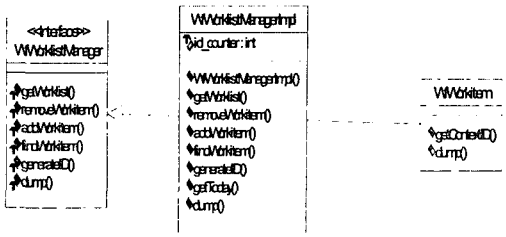

그림 4 WorklistManager 관련 객체

3.4 Context Manager 관련 객체

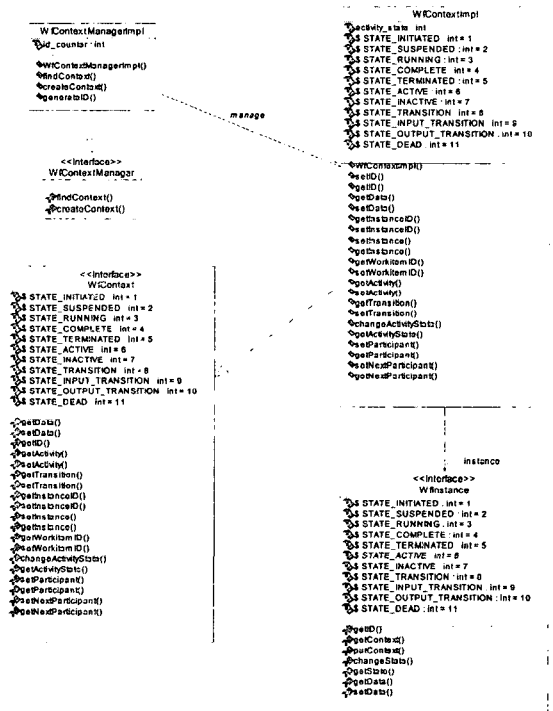

그림 5 ContextManager 관련 객체

그림 5은 WfContextManagerImpl을 기준으로 해서 작성한 클래스구조도이다. WfContextManager는 WfContext를 관리한다. WfContext는 WfInstance와 연결되어서 워크플로우에 대한 데이터를 사용하게 된다. WfContext가 WfInstance에서 관리되지 않고 따로 WfContextManager가 존재하는 것은 실제로 워크플로우 엔진의 Use Case에서 WfInstance보다도 WfContext를 많이 사용하기 때문에 관리 성능과 편의성 면에서 분리하고 또 WfContextManager는 Context의 복제, 통합 등의 기능을 추가로 수행하기 때문이다.

3.5 Notification Manager 관련 객체

WfNotificationManager는 매우 간단한 관리 객체이다. WfWorklistManager와 비슷하게 각 사용자별로 메시지를 주고받는 기능을 지원한다. 사용자는 Notification Manager를 통해서 자신에게 도착한 메시지를 확인하고 다른 사용자로 메시지를 보내는 등의 기능을 수행할 수 있다.

4. 워크플로우 프로세스 스케줄링

워크플로우 엔진에서 가장 중요한 일은 워크플로우 프로세스의 스케줄링이다. 이 작업으로 인해서 워크플로우 엔진이 살아서 움직이게 된다. 워크플로우 프로세스 스케줄링이 하는 일은 워크플로우 인스턴스의 상태관리이다. 인스턴스의 상태에 따라서 워크플로우를 다음 단계로 움직이게 하거나 워크아이템을 추가, 삭제 등의 기능을 하도록하고 컴포넌트 등의 Invoked Application을 실행한다. 워크플로우 인스턴스의 내부에는 두 가지의 상태가 존재한다. 하나는 인스턴트의 자체의 상태이고 다른 하나는 진행 중인 인스턴트에서 현재 수행 중인 액티비티 상태가 있다. 워크플로우 프로세스는 WfInstanceManager를 통해서 주기적으로 스케줄 된다. 여기에서 InstanceManager는 인트턴트 상태 전이 메소드 호출시 워크플로우 프로세스의 상태를 검사해서 워크플로우를 진행시킨다.

4.1 워크플로우 프로세스 상태변화

워크플로우 인스턴스의 상태변화는 그림 6과 같은 형태로 변화한다.

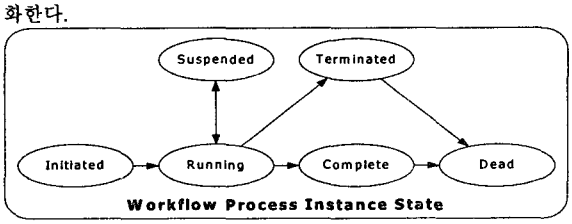

그림 6 Workflow Process Instance State Diagram

Initiated는 워크플로우가 시작된 바로 직후 상태이다. 특별한 동작을 하지 않고 RUNNING으로 바뀌길 기다린다. 워크플로우 정의에 자동으로 RUNNING으로 바뀌도록 지정할 수 있다. Running는 워크플로우가 진행중인 상태이다. Activity의 상태를 검사해서 워크플로우를 진행한다. Process_In_Running_State()안에서 Activity의 상태에 따라서 적절한 동작을 취한다. 워크플로우 엔진의 핵심 부분이다. Suspended는 일지 중지한 상태이다. Terminated는 워크플로우가 비정상 종료된 것이다. 생성한 사람에게 메시지로 비정상종료된 것을 알리게 된다. Complete는 워크플로우가 정상 종료된 상태이다. 이 상태가 될 적에 생성한 사람에게 메시지로 알려준다. Dead는 워크플로우가 완전히 끝나서 Reference와 메모리를 정리하는 단계이다.

4.2 Activity 상태변화

그림 7은 Activity의 상태의 변화를 나타낸다. Initiated는 Activity가 처음 생성된 상태이고 아직 실행은 되지 않은 상태이다. 이 상태에서는 Activity의 'start_mode'를 검사한다. start_mode가 interactive이면 그냥 상태가 변할 때까지 기다린다. start_mode가 automatic이면 start_condition을 검사한다. start_condition이 만족되면 Active상태로 변화한다. Active는 Activity가 실행되기 직전 상태이다. 만약 Activity의 execution_mode가 automatic이면 invoked application을 실행하고 Running상태로 움직인다. execution_mode가 interactive이면 workitem을 생성하고 Running 상태로 움직인다. Running는 Activity가 수행중이고 그 수행이 끝나기를 기다리는 상태이다. Activity의 finish_mode가 automatic이면 finish_condition을 검사한다. 그리고 finish 조건을 만족하면 Complete로 변화한다. Complete는 execution_mode가 interactive이면 먼저 생성했었던 workitem을 제거한다. automatic이면 바로 transition을 처리하는 단계로 넘어간다. Input Transition 단계에서는 지금까지 처리했던 activity로부터 다른 activity로 전이하기 위해서 지금 처리했던 activity를 input activity로 하는 transition을 찾아서 이 transition의 input 조건을 검사한다. 다양한 input type에 따라서 다르게

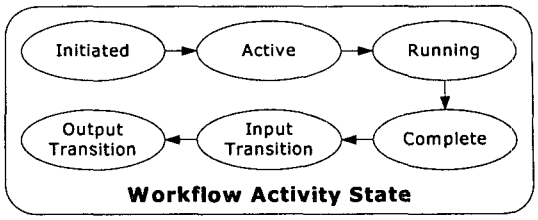

그림 7 Workflow Activity State Diagram

조건 검사를 수행하여야 한다. 만족하면 output transition 단계로 넘어간다. Output Transition 단계에서는 transition의 output 조건을 검사한다. input과 마찬가지로 다양한 type에 따라서 다르게 조건 검사를 수행한다. 만족하면 다음 activity를 Initiated 상태로 바꾸면서 넘어간다.

4. 결론

본 연구에서는 재 사용성이 가능하고 이질적인 정보를 가지는 정보 시스템들간의 상호 호환이 가능한 분산 컴포넌트 방식의 프로그램과, 정적인 업무의 흐름이 아닌 동적으로 업무의 흐름을 변경할 수 있으며 장시간에 걸쳐서 실행되는 프로세스의 지원과 비동기적인 실행을 보장하 워크플로우 워크 플로우 관리 시스템 설계 및 구현에 대해 기술하였다. 특히 워크플로우 관리 시스템의 핵심부분인 Run-Time시의 워크플로우 엔진이 어떤 방식으로 동작하는지에 대해서 설명했다. 워크플로우 엔진의 내부 구조인 Class Diagram을 제시했으며. 엔진에서 중요한 부분인 워크플로우 프로세스 정의를 읽어서 처리하는 부분과 워크플로우 스케줄링에 대한 부분을 설명했다. 스케줄링에서는 워크플로우 프로세스와 Activity의 상태관리에 대해서 주로 기술했다. 이를 통하여 분산 컴포넌트 방식의 재사용성이 가능하며 다양한 이질적인 정보를 가지고 있는 정보 시스템을 통합도 할 수 있으며 B2B에서의 다양한 업무 형태를 지원할 수 있는 Workflow 서비스를 웹에서 구현할 수 있을 것이다. 또한 Java 기반으로 WorkFlow엔진을 구현했으므로 Platform Independence하게 Workflow 서비스를 받을 수 있다.

5. 참고문헌

[1] Workflow Management Coalition, "Workflow Standard Interoperability Abstract Specification" ,WfMC-TC-1012, Oct.1996. (http://www.wfmc.org/standards/docs/if4-a.pdf)

[2] Workflow Management Coalition, "Workflow Reference Model", WfMC-TC-1003, 1.1, 19 Jan 1995. (http://www.wfmc.org/standards/docs/tc003v11-16.pdf)

[3] David Holliingsworth, workflow Management Coalition, "The Workflow Reference Model",http://www.aiim.org/wfmc, Jan.1995

[4] Internet Engineering Task Force(IETF), " Requirements for Simple Workflow Access Protocol", http://www.ietf.org, August, 1998

[5] 배준수, 정석찬, 서영호, 김영호, 강석호, "CORBA 기반 분산 워크플로우 관리 시스템 개발", 대한산업공학회/한국경영과학회 '98 춘계공동학술대회 논문집, 1998

[6] W3C, "Extensible Markup Language (XML)", REC-xml-19980210, 1.0, 10 Feb. 1998. http://www.w3.org/TR/1998/REC-xml-19980210

[7] Thomas Anne, "Enterprise JavaBeans Technology: Server Component Model for the Java Platform", Sun Microsystems Inc., December 1998.

[8] "비즈니스 서버 컴포넌트의 재사용 지원 플랫폼의 수립 및 프로토타입 개발에 관한 연구," 결과보고서, 한국전자통신연구원, 2000년