EJB 기반의 워크플로우 정의 데이터베이스 에이전트 설계 및 구현

An EJB-Based Database Agent for Workflow Definition

오 동 근* 김 광 훈**
Dong-Keun Oh Kwang-Hoon Kim

요 약

본 논문은 워크플로우 관리 시스템의 주요 기능중에 하나인, 워크플로우 모델 정의 기능을 담당하는 EJB 기반의 DB 에이전트를 설계 및 구현함으로서, 이를 EJB 컴포넌트화 하는데 그 목적을 두고 있다. 본 논문에서 구현한 EJB 기반의 DB 에이전트는 빌드타임 클라이언트 각각의 모듈과 DB 사이에 위치하여, DB에 대한 연결관리와 자료의 호출 및 저장을 수행한다. EJB의 장점으로는 분산객체 기술에 기반을 둔 표준 서버측 컴포넌트 모델인 점과 그리고 시스템 장애(failover), 트랜잭션, 보안 등의 기능들을 서버 차원에서 안정적으로 지원하는 기능을 가지고 있다. 이러한 EJB를 워크플로우에 적용함으로서 시스템 이질성 및 상호 운영성의 제한과 급격히 증가하는 프로세스에 따른 시스템 오버헤드 및 장애(failure)에 대한 문제를 해결하여 시스템의 정확성과 신뢰성을 높일 수가 있다.

Abstract

This paper deals with an EJB-based database agent(component) used to define workflow processes, which is a core function of the e-Chautauqua workflow management system that is an on-going research product. We describe about how to design and implement the EJB-based DB agent that is deployed on EJB server as a component. The agent is located between the build-time clients and the database system, and manages database accesses, such as retrieves and stores, from the workflow definition components. Through the EJB technology, we are able to accomplish a stable database agent that can be characterized by the distributed object management, reliable recovery mechanism from system failovers, reliable large-scale transaction management, and the security functions.

1. 서 론

70년대에 언급된 사무 자동화(Office Automation)라는 개념은 사무 업무에 있어 불필요한 문서들을 줄이고 서로 관련된 많은 작업들을 자동화하려는 것이다. 하지만 사무 자동화를 이루려는 많은 노력들은 다른 많은 IT의 움직임처럼 그 초창기의 목적을 완전히 이루지 못하였고, 오히려 문서들의 급격한 증가와 더불어 워드프로세스, 팩스와 같은 독립된 사무 업무들만이 자동화 되었을 뿐이다. 나아가 문서에 의존한 업무 처리와 기업내 비효율적인 프로세스의 사용은, 급격히 변화하고 있는 IT 기술과 더불어 새로운 시대의 요구들을 가로막고 있는게 현실이다[1]. 이러한 시점에서 워크플로우 관리 시스템은 IT의 새로운 분야로서 그 중요성과 함께 점차 부각되고 있다. 최근 B2C (Business To Customer)/B2B(Business To Business)로 대변되는 전자상거래 및 전자시장(E-Market Place)의 활성화가 급속하게 확장됨에 따라, 워크플로우를 기반으로 한 기업 내부 업무 흐름의 자동화를 의미하는 B2E(Business To Enterprise)의 구축을 더욱 가속화시키고 있다. 그리고 워크플로우의 기술 및 시스템은 결국 기업 또는 조직체 내의 모든 업무 처리 절차들을 통합 관리하는 인프라 구조

* 준 회 원 : 경기대학교 대학원 전자계산학과 석사과정
dkoh@kyonggi.ac.kr
** 종신회원 : 경기대학교 정보과학부 교수
kwang@kyonggi.ac.kr

로 인식되고 있다.

현재 수백 개의 상용 제품들이 워크플로우 특성을 반영한 시스템으로 보고되고 있다. 이들 시스템들은 프로세스를 지원하고 개선하는 도구로서 많은 성장을 하고 기술적 진보를 이루었으나, 다음과 같은 측면에서 한계성을 지니고 있다. 첫째, 시스템의 이질성 및 상호 운영성 지원 측면에서의 제한이 있다. 분산 환경에서 동작하는 워크플로우 관리 시스템은 다양한 플랫폼에서 참여하는 사용자를 모두 지원해야 하며, 기타의 정보 시스템(legacy system과 같은 외부 응용 프로그램)과의 통합을 지원해야 하고, 또한 다른 종류의 워크플로우 관리 시스템과의 상호 운영성(Interoperability)이 있어야 한다. 특히 전자 상거래에서 B2B의 경우에는 회사와 회사간의 서로 다른 시스템들을 호환할 수 있어야 한다. 둘째, 장애(failure)에 대한 대응이 부족하여 정확성과 신뢰성이 떨어지는 점이다. 장애가 발생한 경우에 대한 대응은 필수적인 부분임에도 불구하고 대부분의 시스템들이 이 기능에 대한 대비가 부족하다.

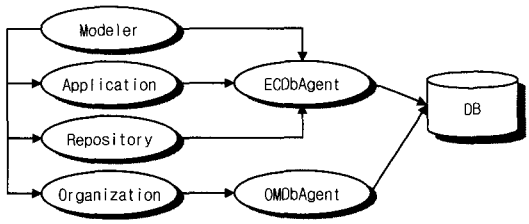
위에 제시된 문제를 본 논문에서는 다음과 같은 EJB 기술의 장점을 가지고 워크플로우에 적용하였다. 첫째, EJB는 분산 객체 기술에 기반을 둔 표준 서버측 컴포넌트 모델이다. EJB 컴포넌트 모델을 이용해 비즈니스 객체를 개발하면, 그것들이 EJB 스펙을 지원하는 어떤 컴포넌트 트랙잭션 모니터(CTM)에서도 실행 가능하다는 것을 뜻한다. 이는 CORBA 기반의 CTM 제품을 검토하는 고객들이 부딪쳤던 가장 큰 문제, 즉 '공급 업체 종속'이라는 공포를 충분히 제거할 수 있을 만큼 강력한 의미를 갖는다[2]. 따라서 자바가 제공하는 유연성과 이식성, CORBA가 제공하는 분산 객체 인프라를 결합함으로써 인터넷 환경하에서 웹 서버를 포함한 다양한 서버들과 자바의 이식성 있는 코드를 기반으로 한 클라이언트를 구축할 수 있다. 둘째, EJB는 보안(Security), 시스템 장애(failover), 트랜잭션(transaction)등의 시스템 레벨의 기능들을 서버 차원에서 안정적으로 지원할 수 있는 장점을 가지고 있다. 그리고 Java의 One-Write, Use Anywhere의 특징을 받아들여 "Write Once, Deploy Anywhere", 즉 한번만 작성하면 EJB Server Spec을 구현한 어떠한 CTM(Component Transaction Monitor)에서도 동작할 수 있는 것이다[3]. 또한 대부분의 EJB 서버는 Container- Managed Entity Bean이라는 것을 지원하여 EJB와 데이터베이스 연동을 아주 쉽게 한다. 이러한 EJB 기술을 워크플로우에 적용함으로서 시스템 이질성 및 상호 운영성의 제한과 급격히 증가하는 프로세스에 따른 시스템 오버헤드 및 장애(failure)에 대한 문제를 해결하여 시스템의 정확성과 신뢰성을 더욱 향상시킬 수 있다.

따라서 본 논문은 워크플로우 관리 시스템의 주요 기능 중에 하나인, 워크플로우 모델 정의 기능을 담당하는 EJB 기반의 DB 에이전트를 설계 및 구현함으로서, 이를 EJB 컴포넌트화 하는데 그 목적을 두고 있다. 다음 2절에서는 워크플로우 시스템의 일반적인 개념에 대해서 언급하고, 3절에서는 DB 에이전트와 연동될 워크플로우의 빌드타임 기능 구분 및 특징들을 기술, 그 구조를 설계한다. 4절에서는 EJB 기술을 적용한 워크플로우 정의 DB 에이전트의 설계 및 구현을 제시하고 마지막으로 결론을 맺는다.

2. 워크플로우 개념

워크플로우(workflow)란 한 조직체 내에서 발생하는 여러 단계의 복잡하고 다양한 비즈니스 업무 흐름을 정의하고 이의 수행을 위한 효율적인 상호 작업 환경을 제공하는 자동화된 서비스를 의미한다. 워크플로우는 비즈니스 프로세스를 나타내는 일련의 물리적 또는 논리적 단위의 엑티비티(activity)들과 이를 수행하기 위한 참여자(participant), 그리고 엑티비티들간에 전달되는 문서 또는 정보들로 구성된다[4,5]. 워크플로우 관리

시스템의 기능은 크게 빌드타임 부분과 런타임 부분으로 나누어진다. 빌드타임 부분은 워크플로우 프로세스를 사용하기 전에 정의하는 기능을 수행하고, 런타임 부분은 설정된 워크플로우 프로세스 정의에 따라 프로세스를 실제로 동작시키는 기능을 수행한다.



(그림 1) DB 에이전트와 빌드타임과의 연결 관계

3. 빌드타임 구조

3.1 빌드타임

비즈니스 프로세스는 하나 또는 그 이상의 분석, 모델링과 시스템 정의 기술에 이용함으로서 실세계로부터 공식적으로 컴퓨터가 수행 가능한 표현으로 변환된다. 결과적인 정의는 때로 프로세스 모델, 프로세스 템플릿(template), 프로세스 정의라고 불린다. 프로세스 정의는 인위적인 정의와 워크플로우의 정의를 포함하는 프로세스의 컴퓨터적인 표현이다. 이러한 프로세스 정의는 이산되어 있는 많은 엑티비티 과정들과 컴퓨터와 인간과의 상호작용에 있어서의 그리고/또는(and/or) 동작, 그리고 다양한 엑티비티 과정을 통해서 진행되는 프로세스의 진행을 통제하는 규칙들로 구성된다. 또한 실행 시 엑티비티에 필요한 프로그램, 엑티비티 간에 주고받는 데이터라든가 수행을 맡게 되는 참여자(participant)나 롤(role)들도 기업내 조직도를 바탕으로 정의를 해야 한다[4].

3.2 빌드타임의 기능 구분

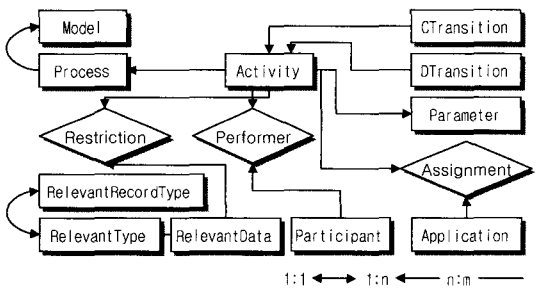
빌드타임 기능의 구분은 크게 다음과 같이 나누어 볼 수 있다. 첫째, 비즈니스 업무 프로세스 정의 기능(Modeler), 프로그램 정의 기능(Application), 데이터 정의 기능(Repository), 그리고 조직 정의 기능(Organization)이다. 다음 그림은 본 논문에서 구현한 DB 에이전트와 빌드타임과의 연동된 그림이다.

Modeler는 비즈니스 업무 프로세스를 모델링하고, 업무 프로세스를 구성하는 각 작업에 대한 정의 및 작업들 간의 연결 관계를 지정한다. Application 모듈은 응용 프로그램을 등록하는 기능을 가지고 있으며, 새로운 응용 프로그램을 등록, 수정, 삭제한다. Repository 모듈은 데이터 타입 및 데이터 구조를 등록하는 기능을 가지고 있으며, 새로운 타입 및 구조를 등록, 수정, 삭제한다. Organization 모듈은 책임자 및 수행자, 역할(role), 조직도를 등록, 수정, 삭제하는 기능을 가지고 있다. 이러한 4개의 빌드타임 모듈은 DB의 호출 및 저장을 두 개의 DB 에이전트(ECDbAgent, OMDbAgent)를 통해서만 DB에 접근한다.

3.3 빌드타임의 ER 다이어그램

본 시스템의 빌드타임 ER 다이어그램은 다음 그림 2와 같다.

부가적으로 하나의 엑티비티는 여러 참여자(Participant)에 의해 실행될 수 있으며, 한 참여자



(그림 2) 빌드타임의 ER 다이어그램

는 여러 엑티비티를 실행할 수 있으므로 엑티비티와 참여자와의 관계는 n:m이다. 또한 참여자는 조직(Organization Unit), 멤버(Member), 롤(Role) 중의 하나가 된다. 그리고 멤버는 여러개의 조직에 속할 수 있으며 한 조직에는 여러 멤버들이 속할 수 있으므로 멤버와 조직과의 관계는 1:m이다. 그리고 멤버는 여러 롤을 가지고 한 롤에는 여러 멤버들이 있을 수 있으므로, 멤버와 롤은 n:m의 관계를 가지게 된다.

4. DB 에이전트의 구조

4.1 EJB 아키텍쳐

EJB 아키텍쳐는 객체지향 분산 엔터프라이즈 애플리케이션의 개발 및 분산 배치를 위한 컴포넌트 아키텍쳐이다. 아래의 그림 3은 EJB 컨테이너의 모델 개요를 설명한다.

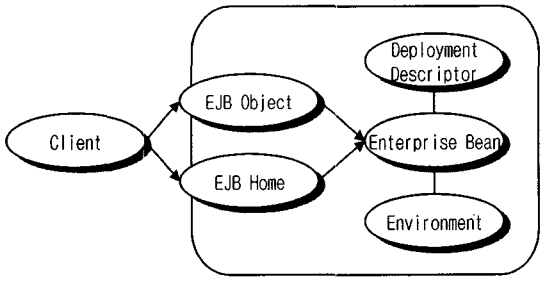
엔터프라이즈 빈을 구현하기 위해서는 EJB 객체, EJB 홈 인터페이스(Home interface), 엔터프라이즈 빈(Enterprise bean)이 필요하다. EJB 객체는 빈이 어떤 일을 수행하기 위해서 외부 세계에 제공하는 빈의 비즈니스 메소드를 정의한다. 그리고 클라이언트의 모든 메소드 호출을 가로채서 배치 디스크립터(Deployment Descriptor) 설정에 기반한 빈에 대해서 트랜잭션, 상태관리, 영속성, 보안 서비스를 이행한다. 홈 인터페이스(Home interface)는 JNDI를 통해 접근 가능하며 새로운 빈을 생성, 제거하고, 찾아내는 등의 빈의 라이프 사이클 메소드를 정의한다[3]. 빈 클래스(Bean class)는 실제로 빈의 비즈니스 메소드를 구현한다.

4.2 EJB의 자원 관리

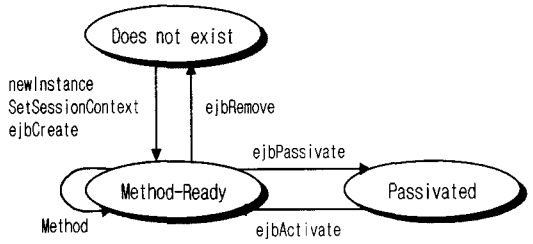
대규모의 시스템에서는 동시에 수천에서 수백만 개까지의 객체를 동시에 사용하게 된다. EJB 서버에서는 이러한 문제점의 해결 방안으로 인스턴스풀링(Instance Pooling)과 스와핑의 두 방법을 사용하고 있다. EJB는 클라이언트에 의해 사용되고 또는 사용되지 않는 과정을 반복하면서 동작하게 되는데, Bean은 클라이언트의 요청이 오고 서비스를 제공하면서 상태를 전이하게 된다. 컨테이너(Container)에서는 Bean의 인스턴스를 생성하고 EJB 객체와 연결하고, 더 이상 사용되지 않는 인스턴스를 삭제하는 등의 과정을 거쳐 Bean의 상태를 변화시킨다.

EJB의 컴포넌트는 크게 엔티티 빈(Entity Bean)과 세션 빈(Session Bean)으로 나누어지는데, DB 에이전트는 스테이트풀(Stateful) 세션 빈을 사용한다. 그림 4는 스테이트풀 세션 빈의 라이프 사이클을 보여준다.

그림 4에서, 처음 상태는 DB 에이전트의 인스턴스가 생성되지 않은 것이다. 그리고 클라이언트의 요청이 생기면 메소드 준비 상태로 전이된다. 그 다음 상태는 빈 인스턴스가 클라이언트로부터 메소드를 서비스하고 있지 않는 비활동 기간인데, EJB는 리소스를 절약하기 위해서 컨테이너는 Bean



(그림 3) EJB 컨테이너의 모델 개요



(그림 4) DB 에이전트의 라이프 사이클

의 대화 상태를 보존한 채 Bean의 인스턴스를 메모리로부터 제거하여 비활성화 시킬 수 있다[6].

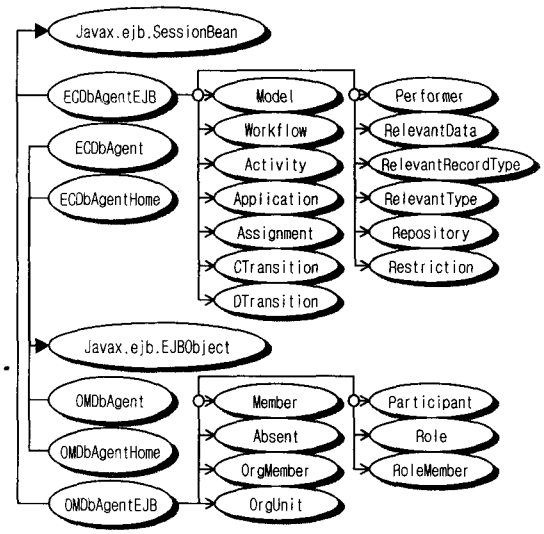
4.3 DB 에이전트의 구현 환경

(표 1) DB 에이전트의 구현 환경

구현 언어	Java 1.3.1
EJB 서버	J2EE 1.2.1
데이터 베이스	Oracle 8.0.5

4.4 DB 에이전트의 클래스 구성

DB 에이전트의 클래스 구성은 다음 그림 5와 같다. EJB 객체, 홈 인터페이스, 엔터프라이즈 빈은 각각 (ECDbAgent, OMDbAgent), (ECDbAgent-Home, OMDbAgentHome), (ECDbAgentEJB, OMDbAgentEJB) 이다. EJB 객체와 홈 인터페이스는 그림 5에서 보는바와 같이 java.rmi.Remote를 계승한 javax.ejb.EJBObject를 상속 받는다. 그리고 빌드 타임의 구성 테이블들을 두 개의 엔터프라이즈 빈의 헬퍼(Helper) 클래스로 구현하였다. 다음은 EJB 객체와 홈 인터페이스의 소스 일부이다.



(그림 5) DB 에이전트의 클래스 다이어그램

```
public interface ECDbAgent extends EJBObject
  public void insertModel(ECDefModel ecdefmodel) throws RemoteException;
  public ECDefModel getModel(String model_id) throws RemoteException;
  public ECDefModel[] getModelAll() throws RemoteException;
  public void deleteModel(String model_id) throws RemoteException;
  public void updateModel(ECDefModel ecdefmodel) throws RemoteException;
  public int how_model() throws RemoteException;
...

public interface ECDbAgentHome extends EJBHome
public ECDbAgent create() throws RemoteException, CreateException;
```

4.5 DB 에이전트의 테이블 구성

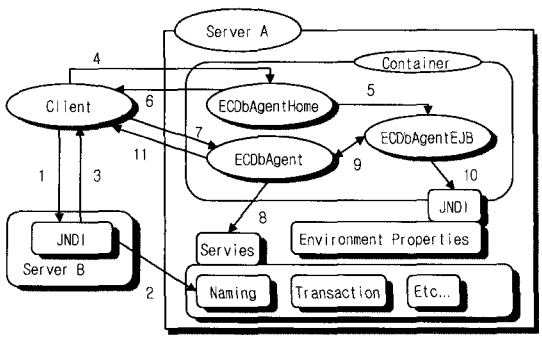
테이블 종류는 그림 5에서와 같이 ECDbAgent에 Model, Workflow등 13개, OMDbAgent의 7개로 구성되어 있다. 그 중에서 Workflow의 테이블 구조는 표 2와 같다.

(표 2) DB 에이전트의 Workflow 테이블

ECDefWorkflow		
model_id	varchar2(63)	모델 식별자
id	varchar2(63)	프로세스 식별자(PK)
name	varchar2(63)	프로세스 이름
description	varchar2(1024)	설명
priority	number	우선 순위
limit	varchar2(10)	예상 수행 시간
duration_unit	number	수행 시간 단위
author	varchar2(63)	작성자
version	varchar2(20)	버전
created	varchar2(19)	작성일자
responsible	varchar2(63)	실행 시간 책임자
valid_from	varchar2(19)	유효 기간 시작 일자
valid_to	varchar2(19)	유효 기간 만료 일자
status	number	프로세스 정의 상태

4.6 DB 에이전트의 내부 동작 과정

그림 6은 ECDbAgent의 내부 동작 과정을 설명하고 있다. 클라이언트는 EJB 서버로 연결을 하고



(그림 6) DB 에이전트의 내부 동작 과정

특정 EJB Home(ECDbAgentHome, OM-DbAgentHome)을 얻기 위해 JNDI를 사용한다. 다음은 연결과정의 구현 소스 일부이다.

```
...
Hashtable props = new Hashtable();
props.put("java.naming.factory.initial","com.sun.jndi.cosnaming.CNCtxFactory");
props.put("java.naming.provider.url","iiop://203.***.***.***:1050");
Context initial = new InitialContext(props);
Object objECDbAgent = initial.lookup("ECDbAgent");
ECDbAgentHome home = (ECDbAgentHome)PortableRemoteObject.narrow(
                    objECDbAgent, ECDbAgentHome.class);
ECDbAgent agent = home.create();
...
```

(1)-(3) InitialContext의 생성자를 통해 프로퍼티에 지정된 EJB 서버를 찾고 lookup() 메소드는 JNDI 서비스에서 ECDbAgent로 바인딩된 ECDb-AgentHome 인터페이스의 레퍼런스를 얻어온다. 다음으로 narrow()를 통해 ECDb-AgentHome 인터페이스를 사용할 수 있게 된다.

(4) 클라이언트는 ECDbAgentHome 오브젝트를 통해서 ECDbAgentEJB 생성을 요청한다.

(5) ECDbAgentHome은 ECDbAgentEJB를 생성한다. 또한 클라이언트의 요청이 있을때는 ECDbAgent-EJB를 제거할 수 있다.

(6) ECDbAgent에 대한 레퍼런스가 클라이언트에게 반환된다.

(7) 클라이언트는 ECDbAgent를 통해 ECDbAg-entEJB를 호출한다.

(8) ECDbAgent는 자신에게 요청된 모든 메소드를 EJB 서비스(Transaction, Persistence, Security...)로 덮는다.

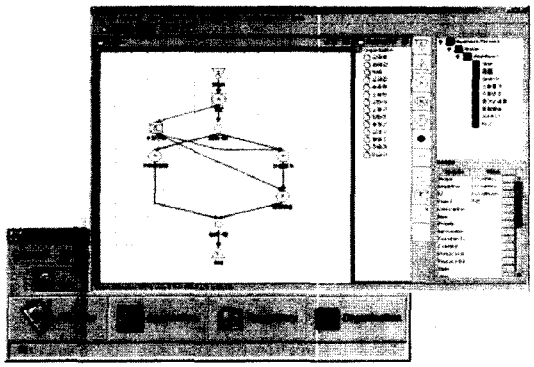
(9) ECDbAgent는 메소드 호출을 ECDbAgent-EJB에게로 전달하고, 그에 대한 반환 값(return value)들을 받는다.

(10) 필요하면 JNDI를 통해 배치 디스크립터(Deployment Descriptor)에 있는 프로퍼티에 접근한다.

(11) ECDbAgent는 비즈니스 호출에 의해 획득한 값들을 클라이언트에게 반환한다[6].

4.7 DB 에이전트와 모델러(Modeler)의 연동

빌드타임 클라이언트의 초기화면은 4개의 모듈(Modeler, Application, Repository, Organization)로 구성되어 있다. 유저는 맨처음 로그인 과정을 거쳐(OMDbAgent 호출) 사용자 인증의 과정을 받게 된다. 그림 7은 유저가 모델러를 ECDb- Agent와 상호 작용하면서 작업하는 과정을 보여주고 있다.



(그림 7) 빌드타임 클라이언트의 모델러 수행 화면

5. 결 론

현 워크플로우 관리 시스템의 큰 단점은 시스템의 이질성 및 상호 운영성을 지원하는 측면에서의 제한이 있고, 급격히 증가하는 프로세스에 따른 시스템 오버헤드, 장애(failure)에 대한 대응이 부족하여 정확성과 신뢰성이 떨어진다는 점이다. 이에 본 논문은 이러한 문제점의 해결 방안으로 분산 객체 기술에 기반을 둔 표준 서버측 컴포넌트 모델인 EJB를 워크플로우에 적용하여, 워크플로우 정의 DB 에이전트를 설계 및 구현을 하였다.

참 고 문 헌

[1] Edward A. Stohr, J. Leon Zhao, "Workflow Automation : Overview and Research Issues", Information Systems Frontiers, Volume 3, Issue 3, pp. 281-287, 2001.

[2] Richard Monson-Haefel, (김동균, 우미영, 강승우 역), "Enterprise Java Beans", pp. 31-40, 한빛미디어, 2000.

[3] 박지훈, 남궁윤, 김종윤, 전부현, "Enterprise Java Beans 1", pp. 5-20, 중앙대학교 자바 동호회 JSTORM, 2000.

[4] David Hollingsworth, "Workflow management Coalition The Workflow Reference Model", pp. 6-9, Workflow Management Coalition, 1995.

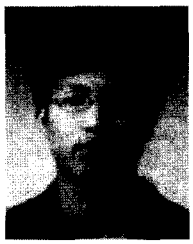
[5] "Workflow management Coalition Terminology & Glossary", p. 10, Workflow Management Coalition, 1996.

[6] "Enterprise JavaBeans Programming", pp. 1/1-3/29, Sun Microsystems, 2000.

[7] "The JAVATM 2 Enterprise Edition Developer's Guide", Sun Microsystems, 2000.

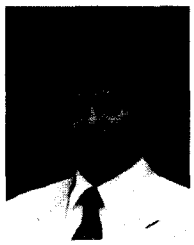
[8] "Interface 1 : Process Definition Interchange Process Model", Workflow Management Coalition, 1999.

◐ 저 자 소 개 ◑



오 동 근

2002년 경기대학교 전자계산학과 졸업예정(이학사)
2002년~현재 : 경기대학교 대학원 전자계산학과(석사)
관심분야 : 워크플로우, 데이터베이스
E-mail : dkoh@kyonggi.ac.kr



김 광 훈

1984년 경기대학교 전자계산학과 졸업(이학사)
1986년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1994년 콜로라도대학교 대학원 컴퓨터과학과 졸업(이학석사)
1998년 콜로라도대학교 대학원 컴퓨터과학과 졸업(이학박사)
1998년~현재 : 경기대학교 정보과학부 교수
관심분야 : 워크플로우, 그룹웨어, CSCW, 분산처리기술, 데이터베이스
E-mail : kwang@kyonggi.ac.kr