韓國 OA學會誌
第8卷 第1號, 2001. 1

원격대학 애플리케이션용 EJB 컴포넌트 추출을 위한 UML 설계에 관한 연구

Efficient UML Modeling Method for Remote University Application EJB Component Extraction

반길우* 최유순* 박종구**

Abstract

EJB application development environment is developing component support Object-Oriented distributed processing, it is component architecture for distributed arrangement. Application developed using EJB is component coupled for business program development easily. EJB is automatically sovled to security, resource pooling, persistency, concurrency, transaction transparency. This paper illustrate for EJB extract to EJB sufficient flexibility its development environment, and it was applicated remote university application domain.

* 원광대학교 대학원 컴퓨터공학과
** 원광대학교 컴퓨터공학과 교수

Ⅰ. 서 론

소프트웨어를 재사용하기 위하여 부품화 하는 방식은 이미 오래 전부터 연구되어 왔다. 최근 들어 그 방식의 하나로 컴포넌트 기반의 소프트웨어 개발 방법이 정착되면서, 컴포넌트 및 CBD는 21세기 소프트웨어 산업을 이끌 핵심적인 방법으로 등장하고 있다. 컴포넌트는 특정 기능을 수행하기 위해 독립적으로 잘 정의된 인터페이스를 가지며 다른 컴포넌트와 조립되어 응용 시스템을 구축하기 위해 사용되는 소프트웨어의 단위를 뜻한다. 특히 컴포넌트는 언제, 어디서, 누구나 필요한 정보를 쉽게 얻을 수 있는 인터넷 환경이 보편화되면서 인터넷상에서 다양한 소프트웨어 부품을 'plug and play' 형태로 조립하여 사용할 수 있음에 따라 컴포넌트 기반 소프트웨어의 개발 추세가 가속화되고 있다.(1)

컴포넌트 기반 소프트웨어가 활성화되기 위해서는 컴포넌트가 운용되는 유연하고 확장성 높은 아키텍처가 필요하다. 선(Sun)사에 의해 제안된 엔터프라이즈 자바빈(Enterprise JavaBeans, EJB) 아키텍처는 컴포넌트 기반의 분산 업무 애플리케이션의 개발과 배치를 위한 컴포넌트 아키텍처이다. EJB를 사용함으로써, 복잡한 분산 객체 프레임웍에 대한 작성없이 확장성과 신뢰성이 높고 안전한 어플리케이션의 작성이 가능하게 되었다.(2) 따라서 EJB는 어떤 회사의 엔터프라이즈 미들웨어에서도 이동성 있고 재사용이 가능한 애플리케이션을 지원하도록 설계되고 있다.

본 논문은 보다 유연성 있는 원격대학 EJB 컴포넌트 설계를 위해서 연구되었다. 2장은 EJB 컴포넌트의 개념과 아키텍처를 설명하고, 3장에서는 프리젠테이션 계층, 비즈니스 로직 계층, 백 엔드 계층으로 계층화한 과정을 UML을 이용하여 다이어그램으로 표현하였으며, 4장에서는 EJB 컴포넌트를 구현하기 위한 빈을 추출하였다. 5장에서는 결론 및 향후 연구 방법을 제시한다.

Ⅱ. EJB 컴포넌트 아키텍처

2.1 컴포넌트 정의

컴포넌트에 대한 정의는 다양하다. Rational사의 Philippe Krutchen은 "컴포넌트는 잘 정의된 아키텍처 상에서 어떠한 기능을 수행하는 시스템의 독립적이면서 대치 가능한 부분이다. 컴포넌트는 인터페이스들의 집합에 대한 물리적인 구현을 제공한다."라고 정의하였으며, Desmond Francis DSouza는 "컴포넌트는 소프트웨어 구현을 갖는 패키지이다"라고 정의하면서 세 가지의 특성을 제시하였다. 즉, 컴포넌트는 독립적으로 개발되고 보급될 수 있어야 하며, 인터페이스를 가져야 하는데 자신이 제공하는 제공 인터페이스와 자신이 필요로 하는 요구 인터페이스를 가져야 하고, 컴포넌트가 지니는 속성을 커스터마이즈할 수 있어야 한다고 제시하였다. 그 외에도 여러 사람이 컴포넌트에 대한 정의를 하고 있는데 공통점을 살펴보면 첫째, 컴포넌트는 모듈화되어 있는 단위이다. 둘째, 컴포넌트는 인터페이스를 가져야 하며 인터페이스를 통해서만 접근이 가능하다. 셋째, 컴포넌트는 구현되어 있는 단위이며 논리적인 모델이 아니라 실제 실행될 수 있도록 만들어진 모듈이다. 넷째, 컴포넌트는 아키텍처를 기반으로 만들어야 한다. 다섯째, 컴포넌트는 사용자가 자신의 목적에 맞도록 컴포넌트의 속성이나 메소드를 변경할 수 있어야 한다는 것이다.

컴포넌트 플랫폼은 컴포넌트 상호간 또는 수행환경에 표현되는 인터페이스를 명세하고, 컴포넌트가 실행시에 서로

통신할 수 있는 기본 규칙 및 서비스 등의 메커니즘을 제공한다.(3) 현재까지의 플랫폼으로는 COM/DCOM, JavaBeans/EJB, CORBA/CCM 등이 있으며, 이들 각각의 특징은 표1과 같다. (4)(5)(6) 이 중 공용 플랫폼을 개발하기 위해 선정한 플랫폼은 EJB(Enterprise JavaBeans)로 확정되었다.

표 1. 컴포넌트 플랫폼 비교표

Comparison Criteria	CORBA	EJB	COM
Cooperation	OMG	Sun Microsystems	MicroSoft
Complexity of Specification	Complex(Extended IDL)	Somewhat complex	The most complex
Comp. Spec. Language	CIDL	None(tools, Java)	IDL
Separation of Public I/F	Separated IDL interface	No separated interface	Separated through interface
Communication Protocols	IIOP	RMI/IIOP, JNDI	RPC
Customizability	Fully Supported(Dynamic binding, Plug-in objects)	The same as left colm, (at Deployment time)	Limited supported(containment, aggregation)
Scalability	Fully scalable and distributable	Fully scalable and distributable	Not supported in Win 2000
Interoperability	Source code level through IDL	Binary level within a specific EJB server	Binary level(Windows platform)
Ruquired Runtime Engines	CIF Engine	EJB container, EJB server	COM Library
Runtime Efficiency	Not yet tested	moderate	Superior for C/S Application
Development Tools/CASE	Not available yet	A few(Web logic, ...)	Most MS Visual Studio
Market Share	Some	A small number	Thousand of products
Market Prediction	Most traditional, academid	Emerging for large-grained components	Greatest market for small-grained component

2.2 EJB 컴포넌트

엔터프라이즈 자바 빈즈는 서버측 컴포넌트 모델의 스펙을 정의하고 있다. Javax.ejb 패키지의 클래스들과 인터페이스들을 사용하여 개발자들은 EJB 스펙에 맞는 컴포넌트를 만들고, 조합하고, 분산 배치할 수 있다. 이에 반하여 자바 빈즈, 즉 자바 API의 java.beans 패키지도 컴포넌트 모델이기는 하지만 EJB같은 서버측 컴포넌트 모델은 아니다. 자바 빈즈가 단일 프로세스 내에서 사용될 목적으로 만들어진 반면, EJB는 복수 프로세스 간에 사용되는 컴포넌트로 디자인 되었다.

자바 빈즈의 컴포넌트가 자바 빈즈 컴포넌트 모델을 따라 만들어진 어떤 GUI 애플리케이션에서도 사용될 수 있는 푸시(push) 버튼이나 스프레드쉬트(spreadsheet) 같은 것이라고 한다면, EJB의 컴포넌트는 고객 비즈니스 객체를 필요로 하는 어떤 비즈니스 애플리케이션을 개발하는 데도 사용할 수 있고, 어떤 EJB 서버에도 배치할 수 있는 Customer 비즈니스 객체라고 할 수 있다.

EJB는 오늘날 가장 진보된 비즈니스 애플리케이션 서버인 컴포넌트 트랜잭션 모니터(CTM)를 위한 컴포넌트 모델이다. EJB는 보안, 리소스 풀링(resource pooling), 퍼시스턴스(persistence), 동시성 제어(concurrency), 트랜잭션 무결성(transaction integrity) 등 비즈니스 시스템이 갖추어야 할 요소들을 자동적으로 해결해 준다.

2.3 EJB 컴포넌트 구조

EJB는 엔티티 빈(entity bean)과 세션 빈(session bean)의 두 가지의 유형이 있다. 엔티티 빈 모델의 개념은 명사에 비유될 수 있다. 엔터프라이즈 빈을 디자인할 때는 두 개의 인터페이스와 클래스를 정의해야 한다. 리모트 인터페이스는 빈이 어떤 일을 수행하기 위해서 외부세계에 제공하는 빈의 비즈니스 메소드를 정의하며, 이 메소드를 구현한

객체를 EJB 객체라고 부른다. 홈 인터페이스는 새로운 빈을 생성하고, 제거하고, 찾아내는 등의 빈의 라이프 사이클 메소드를 정의하며, 이 메소드를 구현한 객체를 EJB 홈이라고 부른다. 빈 클래스는 리모트 인터페이스에 정의된 메소드들에 대응하는 메소드를 가지며, 엔티티 빈에만 필요한 프라이머리 키는 데이터베이스로의 포인터를 제공하는 매우 단순한 클래스이다.

그러나 인터페이스나 클래스에는 런타임 시에 빈이 어떻게 관리되는지에 대한 정보는 없다. 빈이 보안, 트랜잭션, 네이밍, 그리고 기타 분산 객체 시스템에서 제공되는 일반적인 서비스들을 어떻게 받게 되는지는 EJB CTM 서버에 의해 자동으로 관리된다. EJB 서버는 런타임 시에 어떻게 기본적인 서비스들이 각각의 빈 클래스에 적용되는지에 대한 정보를 가지고 있어야 한다. 이를 위해서 사용되는 것이 배치 디스크립터(deployment Description)라고 부르는 클래스의 집합이다. 빈 클래스와 빈 인터페이스가 정의될 때, 빈에 대한 배치 디스크립터가 생성되고, 빈에 대한 데이터로 채워진다.

EJB 객체는 서버측에서 빈의 리모트 인터페이스를 구현하는 분산 객체이다. 즉 서버측에서 빈 인스턴스를 감싸고 EJB 객체 행위를 포함하여 그 기능을 확장한다. EJB 객체는 빈 클래스와 배치 디스크립터에서 제공되는 정보에 기반을 두고 EJB 컨테이너 공급 업체가 제공하는 유틸리티에 의해 생성된다.

EJB 홈은 컨테이너에 엔터프라이즈 빈을 설치할 때에 자동적으로 생성되는 클래스이다. 이것은 홈 인터페이스에서 모든 메소드를 구현하고, 컨테이너가 빈의 라이프 사이클을 관리하는 것을 도와 준다. 컨테이너는 빈과 서버 사이에 동일한 인터페이스를 제공하며 리모트 인터페이스와 홈 인터페이스를 구현해준다. EJB 홈은 EJB 컨테이너와 밀접하게 동작하면서 엔터프라이즈 빈을 생성하고, 제거하고 위치시키는 책임을 진다. 이는 EJB 서버의 자원 관리자, 인스턴스 풀링, 그리고 퍼시스턴스 메커니즘 등 개발자에게 감춰진 부분들과도 상호 관련을 맺고 동작한다. 그림 1은 각각 홈 인터페이스와 리모트 인터페이스를 구현한 EJB 홈과 EJB 객체를 가진 EJB의 구조를 나타낸다. 여기서 빈 클래스는 EJB 객체에 의해 감싸인 형태로 보여지고 있다.(7)

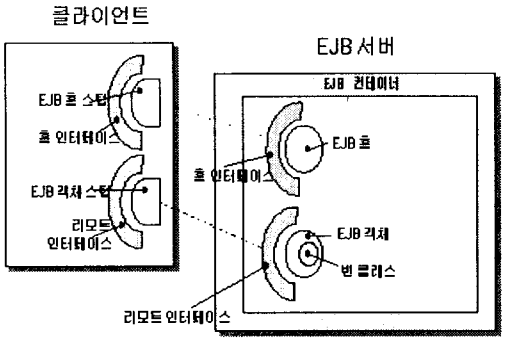

그림 1. EJB 아키텍처

Ⅲ. EJB 컴포넌트 모델링

본 논문에서는 사이버 세계에서 진행되고 있는 원격 교육 업무를 3단계로 구분하여 효율적인 모델링 기법을 제안한다. 그 1단계는 요구사항 분석 단계로 유스케이스 다이어그램을 작성한 후 도메인을 패키지 영역으로 분류하기 위하여

좀더 상세하게 유스케이스 리스트를 작성한다. 요구분석 단계의 결과물에 근거하여 작업장소와 업무를 분야별로 분류하여 시스템 설계를 위한 패키지 다이어그램을 작성하고, 패키지간의 순서를 바탕으로 순서 다이어그램을 작성한다. 2단계는 계층화 설계 단계로서 순서 다이어그램에 준해서 표현 계층(Presentation tier), 비즈니스 로직 계층(Business logic tier), 백 엔드 계층(Back end tier)으로 계층별 객체를 모델링 한다. 비즈니스 로직 계층은 다시 분류되어 빈을 추출하고 빈의 유형을 설계하는데 이용된다. 3단계는 빈의 상세 설계 단계로서 빈의 유형별 인터페이스를 상세하게 설계하고, 인터페이스에 의한 클래스를 설계한다. 이러한 클래스들은 Rational Rose를 이용하여 클래스 다이어그램으로 표현되며 그에 적합한 스켈리튼 코드를 생성한다. 이와 같이 단계별 프로토타입에 의한 UML 모델링 기법은 재사용이 가능하고, EJB 컴포넌트의 효율적인 설계 및 코드 자동생성으로 컴포넌트의 개발 생산성을 향상시켜 준다.

3.1 요구사항 분석

요구사항 분석 단계에서는 실제 시스템을 사용할 사용자로부터 사용자가 원하는 요구사항을 조사하여 문서화한다. 업무 과정을 파악하고 도메인을 분석하며 연동 시스템을 확인할 수 있어야 하고 또한 시스템 요구사항이 파악될 수 있도록 하기 위해서이다. 이를 기반으로 그림 2와 같은 유스케이스 다이어그램을 작성한다. 유스케이스는 도메인 프로세스를 완료하기 위하여 시스템의 액터가 수행하는 순차적인 행위를 설명한 서술식 문서이다. 유스케이스 다이어그램은 일련의 유스케이스, 액터, 유스케이스와 액터간의 관계를 표현한 그림이다. 요구사항 기술서(Requirement Specification)는 유스케이스 모델링 과정에서 가장 중요한 기초 자료가 된다. students는 원격대학에 접속하기 위해서 login을 하고 password를 입력한다. login이 되었으면 subject에서 과목을 선택하여 수강신청을 한다. 신규 students의 경우 회원가입이나, 기존 회원의 정보 수정이 필요시 이름, 주소, 주민등록번호, 직업, 연락처 등을 입력해야 한다. 수강 신청한 전체 과목에 대한 내용과 견적 금액이 표시되도록 하기 위해서 quotes를 실행할 수 있다. 작업이 끝나면 logout을 한다. 이와 같이 객체와 객체간의 행위를 유스케이스 다이어그램으로 모델링 함으로써 사용자의 요구사항을 쉽게 분석하고 설계에 관련된 중요한 정보나 개념을 확보하여 도메인 분류를 쉽게 할 수 있다.

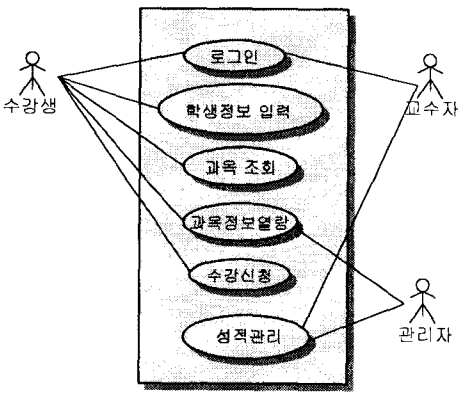

그림 2. 유스케이스 다이어그램

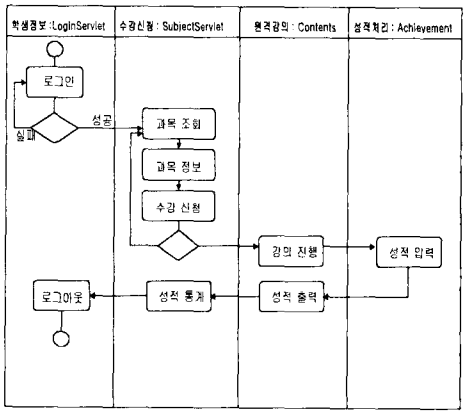

그림 3. 활동다이어그램

그림 3에서는 각각의 작업 흐름에 대하여 활동 다이어그램을 나타냈다. 활동 다이어그램은 시스템의 동적인 사항을 모델링하는데 사용된다. 시스템의 동적인 부분을 모델링할 때 두 가지의 방식으로 활동 다이어그램을 표현할 수 있다. 첫째, 작업 흐름을 모델링하기 위하여 사용하는 경우 시스템과 협동하는 액터의 관점에서 보는 활동에 초점을 두어야 한다. 둘째, 오퍼레이션을 모델링하기 위하여 사용하는 경우 처리 과정의 세부 사항을 모델링하기 위하여 활동 다이어그램을 플로차트로 사용한다. 그림 3은 전자에 해당하는 다이어그램이다.

3.2 시스템 설계

시퀀스 다이어그램은 협도 다이어그램과 함께 시스템의 동적 구조, 즉 객체와 객체그룹 사이, 객체와 객체사이, 객체그룹과 객체그룹사이의 동적인 행위를 기술하게 된다. 시퀀스 다이어그램은 종좌표축으로 시간개념을 도입하고 횡좌표축으로 객체들을 나열하여 그 사이의 상호작용을 표시하였다. 그림 4는 시퀀스 다이어그램을 나타냈다.

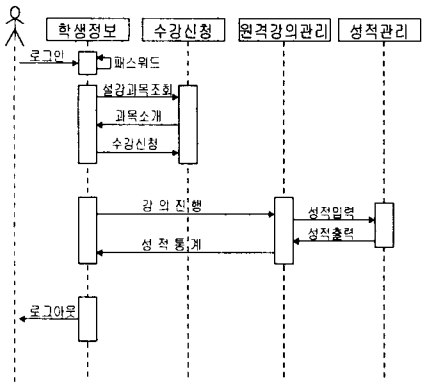

그림 4. 시퀀스 다이어그램

도메인 객체는 시스템의 핵심적인 개념과 행위를 정의한다. 그러나 대부분의 시스템은 도메인 객체뿐만 아니라 다른 많은 서브시스템을 포함하고 있다. 따라서 일반적으로 시스템은 사용자 인터페이스와 영구적인 저장 메커니즘과도 연결되어야 한다. 사용자 인터페이스와 데이터 저장을 포함하는 정보 시스템에 대한 일반적인 아키텍쳐는 3-계층 아키텍쳐(3-Tier Architecture)로 알려져 있다. 그림 5는 EJB 컴포넌트를 모델링하기 위한 첫 단계로서 3-계층별 객체를 추출하여 객체간의 관계를 그림으로 표현하였다. 여기에서 추출된 객체는 좀 더 높은 수준의 단위로 묶어 패키지로 표현된다.

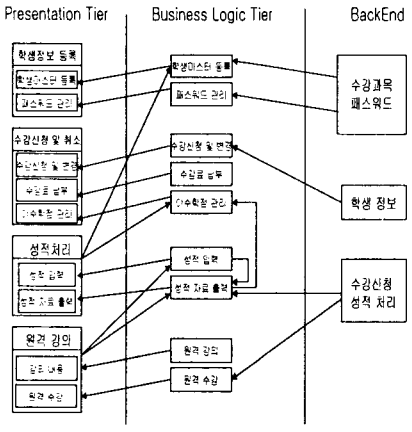

그림 5. 계층별 객체 모델링

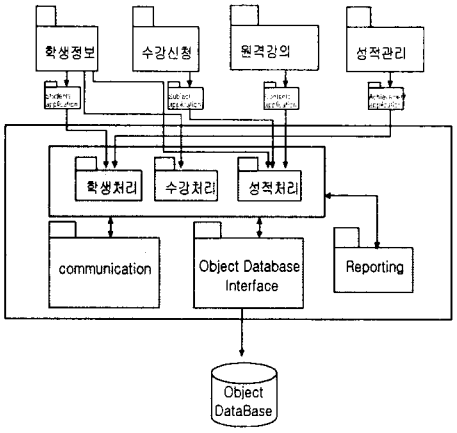

그림 6. 패키지 다이어그램

그림 6은 본 시스템에 존재하는 여러 공통적인 패키지의 보다 상세한 분할과 이러한 패키지간의 종속 관계를 보여주는 패키지 다이어그램을 나타내고 있다. 프로세스 로직 패키지는 프리젠테이션 패키지와 어떠한 종속 관계나 결합을 갖지 않는다는 것을 알 수 있다.

Ⅳ. 빈 추출

엔티티 빈은 현실세계의 객체를 나타내며 , 이런 객체들은 일반적으로 데이터베이스에 저장되는 레코드에 해당된다. 본 논문에서는 패키지 다이어그램을 기반으로 엔티티 빈을 추출하였는데 빈의 종류와 각 빈의 유형은 표 2와 같다.

세션 빈은 클라이언트 애플리케이션의 확장으로, 프로세스나 태스크를 관리하는 역할을 한다. 세션 빈은 강의과목에 대한 수강신청을 하기 위하여 엔티티 빈을 이용한다.는 세션 빈을 추출하고 각 빈의 유형을 나타낸다.

추출된 빈을 상세 설계하기 위한 단계로 클라이언트와의 상호작용이나 다른 빈과의 상호작용을 위한 인터페이스 메소드를 정의하였다. 이러한 인터페이스는 빈의 클래스와 메소드의 추출과정으로 추상화한 데이터들을 통하여 결정한다. 표 3은 컴포넌트 인터페이스의 설계를 나타내고 있다.

표 2. 빈 추출 및 유형 설계

빈	포함된 데이터	빈 유형
students	학생ID, 성명, 주소, 주민등록번호, 직업, 연락처	엔티티
password	암호 입력, 확인, 변경	엔티티
subject	과목code, 과목명, 수강금액, 수강기간, 강사ID	엔티티
prices	수강료 계산	무상태 세션
points	수강신청학점, 이수학점 계산	무상태 세션
achievements	성적 입력, 성적 산출	유상태 세션
remoteach	원격 강의 내용 입력 및 수정	엔티티
teachers	강사ID, 암호, 성명, 주소, 주민번호, 직업, 연락처	엔티티

표 3. EJB 컴포넌트 인터페이스 설계

인터페이스	메소드 명	내 용
Students	StudentsBean()	학생정보 추상화
	getName()	수강 이름 조회
	setName()	수강 이름 등록
	getPassworld()	암호 조회
	setPassworld()	암호 등록
	getAddress()	주소 조회
	setAddress()	주소 등록
	setStudentsContext()	수강생 엔티티 항목 등록
	unsetStudentsContext()	수강생 엔티티 항목 삭제
Subject	SubjectBean()	과목정보 추상화
	setSubject()	과목명 등록
	getSubject()	과목명 조회
	setSubjectContents()	과목 엔티티 항목 등록
	unsetSubjectContents()	과목 엔티티 항목 삭제

엔터프라이즈 빈을 구현하기 위해서 두 개의 클래스와 인터페이스를 정의하였다. 리모트 인터페이스에서는 어떤 일을 수행하기 위해서 외부 세계에 제공하는 빈의 비즈니스 메소드를 정의하였다. 리모트 인터페이스는 java.rmi.Remote를 계승한 javax.ejb.EJBObject를 상속받는다. 홈 인터페이스에서는 새로운 빈을 생성하고, 제거하고, 찾아내는 등의 빈의 라이프 사이클 메소드를 정의하였다. 이는 java.rmi.Remote를 계승한 javax.ejb.EJBHome를 상속받는다. 빈 클래스에서는 실제로 빈의 비즈니스 메소드를 구현하였다. 이 때 빈 클래스는 리모트 인터페이스에 정의된 메소드들에 대응하는 메소드들을 가지고 있다. 엔티티 빈은 javax.ejb.EntityBean을 상속받고, 세션 빈은 javax.ejb.SessionBean을 상속받는다. 엔티티 빈과 세션 빈은 둘 다 javax.ejb.EnterpriseBean을 상속받은 것이다.

V. 결론 및 향후 연구 과제

본 논문에서는 EJB 환경에 보다 유연성 있도록 하기 위하여 기능별 인터페이스 위주로 설계하므로서 빈을 세션 빈과 엔티티 빈으로 구분하였다. 이는 응집도를 높이고 결합도를 낮아지게 하므로 컴포넌트와 빈의 독립성에 의한 컴포넌트의 조립이 쉽고 용이해진다. 따라서 컴포넌트의 결합성을 향상시킬 수 있게 되었다. 또한 잘 정제된 빈은 정형화되어 단순하고, 일괄적이며, 재사용 가능하므로써 네트워크 트래픽, 대기시간, 자원 낭비 등의 성능을 향상시킬 수 있다. 그러나 모델링 과정에서 UML에서 제공하는 모델 요소중에 EJB 컴포넌트에 대한 통일된 표현이 제시되고 있지 않기 때문에 다이어그램을 확장 개발하기 위한 연구가 필요하다고 생각한다.

참고문헌

[1] 김명준,김채규,양영종, "컴포넌트 산업 활성화 방안", 정보처리학회지, July 2000 Vol 7. No 4.

[2] 김수동, "Enterprise JavaBean(EJB) 기반의 컴포넌트 프로그래밍", 정보처리학회지, July 2000 Vol 7. No 4.

[3] 한국전자통신연구원, "공용 컴포넌트 플랫폼 선정을 위한 공청회자료", 한국전자통신연구원, 2000년 3월.

[4] 한국전자통신연구원, "The Component Object Model Specification, v0.9", Microsoft, Oct 24, 1995.

[5] 한국전자통신연구원, "Enterprise JavaBeans Specification v1.1, Public Release 2", Sun Microsystems, Oct 18, 1999.

[6] 한국전자통신연구원, "CORBA Component : OMG TC Document-orbos/ 99-02-05", OMG, March 1, 1999.

[7] Richard Monson-Haefel, "Enterprise JAVABEANS", O'Reilly, March, 2000.

[8] 류형규, 이순천, 류시원, 신성호, "UML 기반 객체지향 클라이언트/서버 구축", 2000.