工學碩士 學位論文

指導敎授 김 일 민

분산 객체 지향 환경에서의 교육시스템 설계 및 구현

Design and Implementation of Education System on distributed Environment

2003 年

한성대학교 컴퓨터 신기술 대학원

컴퓨터 신기술 학과

컴퓨터 공학 전공

兪 在 虎

工學碩士 學位論文

指導敎授 김 일 민

분산 객체 지향 환경에서의 교육시스템 설계 및 구현

Design and Implementation of Education System on distributed Environment

위 論文을 工學碩士 學位論文으로 提出함

2003 년 6월

漢城大學校 컴퓨터 신기술 大學院

컴퓨터 신기술 學科

컴퓨터 工學 專攻

俞 在 虎

유재호의 工學碩士 學位論文을 인정함

2003 年 6 月 日

심사위원장 정 인 환 (인)

심사위원 김 일 민 (인)

심사위원 조 세 홍 (인)

목 차

요 약

1. 서 론 1

2. 관련 연구 3

2.1 분산 컴퓨팅 5

2.2 2-tier 클라이언트/서버(2-tier Client/Server) 5

2.3 Remote Procedure Call(RPC) 7

2.4 Common Object Request Broker Architecture(CORBA) 9

2.5 Remote Method Invocation(RMI) 11

2.6 Enterprise Java Beans(EJB, J2EE) 13

3. 교육시스템 분석 및 설계 15

3.1 교육시스템의 주요 기능 및 요구사항 분석 15

3.2 UML을 이용한 시스템 분석 16

3.2.1 Use Case 모델링(사용자 관점에서의 시스템 분석) 17

3.2.2 Class Diagram 19

4. 교육 시스템 구현 23

4.1 교육 시스템의 구현 23

4.2 교육시스템 에플리케이션 GUI구축사항 25
4.3 Socket, RMI, EJB 환경상에서의 구현 29
4.3.1 Socket을 기반으로 구현한 강의 보드 29
4.3.1.1 DistributedListServer 클래스 31
4.3.1.2 DistributedListClient 클래스 32
4.3.2 Remote Method Invocation(RMI)를 기반으로 한 강의보드 33
4.3.2.1 RList 클래스 34
4.3.2.2 RListImpl 클래스 34
4.3.2.4 RMIList 클래스 35
4.3.3 Enterprise Java Beans(EJB)를 기반으로 한 강의 보드 37

5. 결론 및 향후 과제 39

참고 문헌 40

Abstract 41

그림 목차

그림 2.1. 2-tier 클라이언트/서버 구조.....5

그림 2.2 RPC 클라이언트/서버 구조.....7

그림 2.3 ORB의 대략적 구조.....8

그림 2.4 Remote Method Invocation(RMI)의 개요.....10

그림 2.5 Remote Method Invocation(RMI)의 구조.....11

그림 2.6 네이밍 레지스트리(Naming Registry).....11

그림 2.7 J2EE 서버와 컨테이너.....12

그림 2.8 EJB 컨테이너의 서비스.....13

그림 3.1 교육 시스템의 Use Case Diagram1.....16

그림 3.2 교육 시스템의 Use Case Diagram2.....17

그림 3.3 강의보드 객체 GUI Class Diagram.....19

그림 3.3 강의보드 객체 통신 Class Diagram.....20

그림 3.3 리스트 자료구조의 개요.....21

그림 4.1 강의보드 초기 형태.....23

그림 4.2 강의 게재 및 진행 상황.....24

그림 4.3 Socket/Client 기반의 분산 리스트 구조.....28

그림 4.4 RMI기반의 네트워크 구조.....32

그림 3.5 EJB를 통한 강의보드의 설계 36

표목차

표 3.1 각각의 ACTOR의 역할.....17

표 4.2 socket 기반의 분산 리스트 프로토콜.....29

요 약

컴퓨터 네트워크가 급속하게 발전함에 따라, 이를 이용한 원격교육에 관한 연구가 활발하게 진행되어 왔다. 또한 컴퓨터를 이용함으로써 다양한 멀티미디어 등을 통해 좀더 효과적인 교육을 가능케 한다.

원격교육은 컴퓨터 네트워크를 응용한 분야들 중에서도 각광 받는 분야 중의 하나가 되었으며, 많은 연구들을 통해 다양한 원격교육 시스템이 제안되었다. 하지만 기존에 제안된 원격교육 시스템들은 대부분 클라이언트/서버 구조를 택함으로서 확장성과 속도 면에서 많은 문제점이 노출되었다.

본 논문에서는 이러한 기존 시스템들의 문제점을 해결하기 위해 분산환경 상에서 구현한 새로운 형태의 원격교육 시스템을 제안하였으며, 기존 클라이언트/서버 구조 보다 더욱 향상된 시스템을 제안한다.

1. 서 론

원격교육은 교육자와 피교육자 사이에 다양한 교습매체의 활용과 1대 1 수업을 병행함으로써 교육의 질적 향상을 최대화하려는 제반 활동을 의미한다.

초창기의 원격교육은 우편제도를 기반으로 한 인쇄매체 중심의 형태였다. 1960년대 말, 라디오, 텔레비젼 등의 매체가 등장하면서 이 매체들을 이용한 원격교육이 발달하기 시작하였으며, 이 후 컴퓨터 환경의 급속한 변화와 발전을 토대로 이를 교육환경에 적용 하고자 하려는 연구가 시작되었다. 컴퓨터 네트워크를 이용한 원격교육은 지금도 활발히 연구가 진행되고 있으며, 또한 각종 네트워크 응용 분야들 중에서도 각광받는 분야 중 하나로 자리잡게 되었다.

네트워크 환경에서의 원격교육은 기존의 교육체계 보다 시간적, 공간적 제약을 상당부분 극복할 수 있을 뿐 아니라 다수의 대중에게 효율적이며 경제적인 교육서비스를 제공할 수 있는 점이 큰 장점이라고 할 수 있다. 또한 다양한 멀티미디어 요소의 추가 또한 교육서비스의 질을 한 단계 끌어올린 점도 간과 할 수 없는 부분이다.

기존 네트워크상의 원격교육들 대부분은 미리 만들어진 다양한 매체들을 사용자들이 서버로부터 다운로드 받아 학습을 진행하는 형태가 주를 이루었다. 이러한 형태의 원격교육은 비동기식 온라인 원격 교육(on-line asynchronous distance learning)이라고 통용되고 있고 지금까지도 많은 원격교육 시스템들이 이러한 방식을 취하고 있다. 하지만 비동기식 온라인 원격 교육 시스템들의 가장 큰 문제는 교수자와 수강자 사이의 상호작용(interaction)이 매우 취약하다는 단점이 있는 것이 사실이다. 이러한 단점을 보완하기 위한 노력으로 수강생과 교수자사이의 실시간 상호작용(Real-Time Interaction)이 가능한 동기식 원격교육(on-line synchronous distance learning)시스템의 연구가 활발하게 진행되고 있으나, 현실적으로 여러 한계점이 있는데 구체적 문제는

다음과 같다.

지금까지의 많은 연구들에서 다양한 동기식 온라인 원격교육 시스템이 제안되었으며, 기존 연구 특징은 기능상으로는 서로 다른 특성이 있으나, 가장 큰 문제로 지적되는 점은 구조상으로 클라이언트/서버 방식을 택하고 있다는 점이다.

클라이언트/서버 방식의 장점은 사용자간의 모든 통신이 항상 서버를 통해 이루어지기 때문에 사용자간의 동기를 유지하기는 쉬우나 사용자가 증가함과 동시에 서버의 응용프로그램이 추가되면 많은 부하가 유발되며 따라서 통신 속도의 저하가 가장 큰 문제로 지적되고 있다. 또한 클라이언트 서버 시스템의 가장 큰 문제인 시스템확장성이 어려운 점이 큰 장애로 대두되는 면도 해결해야 할 과제라고 할 수 있다. 또한, 접속자 상호간의 상호작용에 그다지 큰 효과가 없으므로 교육적 효과가 상당부분 반감되는 점도 큰 문제라고 할 수 있다.

이러한 제반적 문제를 해결하고자 본 논문에서는 분산환경을 기반으로 한 새로운 교육시스템을 제안한다.

분산환경이란 기본적으로 네트워크로 연결된 다수의 이 기종 컴퓨터가 마치 하나의 가상 컴퓨터로 인식하여 소프트웨어를 개발 할 수 있는 환경으로서, 궁극적으로 각 사용자가 자신의 개인 컴퓨터 내의 가상 컴퓨터 상에서 응용 소프트웨어를 개발하고 또한 이를 실제적으로 실행시킬 수 있는 환경을 의미한다. 분산 환경은 여러 가지 이점이 있는데, 이러한 이점은 다음과 같다. 첫째, 저비용으로 환경을 조성 할 수 있다. 둘째, 개개의 시스템이 약결합 형태를 취하기 때문에 어느 한 시스템이 고장을 일으키더라도 이 점을 보완할 수 있는 형태로 만들 수 있다. 셋째, 특정 작업을 네트워크에 연결된 다른 컴퓨터들끼리 분산처리 시킬 수 있으므로 작업 효율이 매우 좋다.

본 논문은 이러한 분산환경 하에서 유용한 언어중 하나인 RMI와 엔터프라이즈 자바 빈을 이용해서 기존의 클라이언트/서버 구조의 근본적 문제점을

해결하였으며, 또한 네트워크에 접속한 사용자들끼리 서로 공동 작업이 가능한 응용프로그램을 구현해, 교육시스템내의 교수와 학생간의 상호작용을 상당 부분 높인 점을 밝힌다.

2. 관련연구

2.1 분산 컴퓨팅(Distributed Computing)

컴퓨터 하드웨어 기술이 발전하면서 일반 컴퓨터의 성능이 점차 강력해지고, 가격이 싸지기 때문에 고비용의 중앙 집중형 컴퓨팅환경에서 분산 컴퓨터 환경으로 점진적으로 변경되어 왔다. 중앙 집중형 컴퓨팅은 메인 프레임과 같은 대형 컴퓨터를 주로 사용하였으며, 상당히 고가 시스템이기 때문에 도입, 유지 보수 및 상향 업그레이드가 어려운 문제점이 있다. 반면에 일반 개인용 컴퓨터와 소형 워크스테이션의 성능은 강력해지면서 저 비용이라는 장점이 있다. 때문에 기존에 대형 컴퓨터가 수행했던 작업을 여러 대의 소형 컴퓨터에서 수행하도록 하려는 시도가 분산 컴퓨팅의 원동력이 되어왔다. 분산 컴퓨팅은 오랜 세월에 걸쳐 여러 단계로 발전되어 왔다. 다음의 내용은 개략적인 분산 컴퓨팅의 단계별 발전 상황이다.

2.2 2-tier 클라이언트/ 서버 (2-tier Client/Server)

클라이언트 / 서버라는 용어는 1980년대 개인용 컴퓨터가 네트워크에 연결되어 사용되면서 나타나게 되었고, 1980년대 말 실질적으로 클라이언트/서버 모델이 받아들여지게 되었다. 1990년대에는 개인용 컴퓨터에서 LAN을 이용한 클라이언트/서버프로그램이 본격적으로 도입되었다. 특히, 개인용 컴퓨터의 클라이언트 프로그램들이 GUI형태로 변경되는 시기였으며, 점차 많은 사용자들이 일관된 데이터를 사용하기 위해서 파일서버들이 점차 RDBMS형태로 바뀌게 되었다. 이러한 상황을 토대로 2-tier 클라이언트/서버 구조가 널리 받아들여지게 되었다.

2-tier 클라이언트/서버 형태의 분산 컴퓨팅은 주로 TCP/IP를 이용한 네트

워크 프로그램을 이용해서 작성한다는 특징이 있다. 클라이언트는 주로 개인용 컴퓨터에서 비주얼베이직(Visual Basic), 델파이(Delphie), 파워 빌더(Power Builder) 등을 이용해서 작성되며, 서버는 주로 유닉스 워크스테이션이나 메인 프레임에서 C 언어를 이용해서 작성하게 된다. 2-tier 클라이언트/서버는 다음 그림 2.1과 같은 형태로 구성된다.

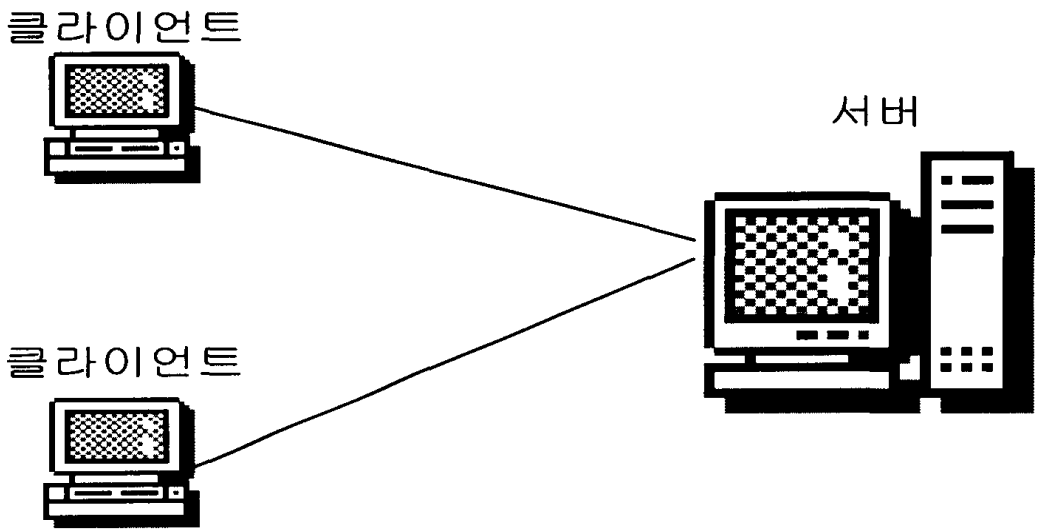

그림 2.1 2-tier 클라이언트/서버 구조

2-tier 클라이언트/서버 구조의 가장 큰 목적은 강력한 서버와 다양한 GUI를 갖게 되는 클라이언트를 분리하기 위한 구조이다. 2-tier 클라이언트/서버 구조는 발전하면서 팻 클라이언트(Fat Client)와 팻 서버(Fat Server) 형태로 발전한다. 팻 클라이언트는 클라이언트 프로그램에 GUI를 위한 프로그램코드 이외에 비즈니스 로직을 위한 프로그램 코드를 많이 추가한 것을 의미한다.

기본적으로 이러한 2-tier 형태를 기반으로 한 클라이언트/서버 구조는 몇 가지 문제점들을 수반하게 된다. 우선 성능적인 측면에서 클라이언트 컴퓨터가 서버 컴퓨터에 상대적으로 성능이 떨어지기 때문에 위에서 언급한 비즈니스 로직을 수행하기에는 어려움이 있다. 반면에 서버측은 동시에 많은 클라

이언트들이 통신을 함으로서 서버에 많은 작업이 요구되기 때문에 서버에 많은 로드(load)가 걸리게 된다.

또 다른 문제로 지적되는 사항은 유지보수의 어려움, 각 개발환경이 서로 다른 문제로 인한 포팅상의 어려움, 특히 확장성이 부족하다는 점등이 큰 문제로 제기 되고 있다.

2.3 Remote Procedure Call(RPC)

네트워크로 연결된 클라이언트와 서버가 함수 호출 형태로 통신 할 수 있는 방법으로 기존의 2-tier 구조에서의 번거로운 네트워크 프로그램 작성을 제거하기 위해 개발되었다.

Remote Procedure Call(RPC)은 서버에서 비즈니스 로직을 C 프로그래밍 언어 모듈로 작성하고, 네트워크로 연결된 클라이언트에서 이 모듈을 호출할 수 있는 방법을 제공한다. RPC는 비즈니스 로직과 데이터 엑세스 기능을 여러 개의 컴퓨터에 분산시켜 수행할 수 있는 방법을 제공하기 때문에 2-tier 클라이언트/서버 구조 보다 확장성이 뛰어나다는 장점이 있다. 또한 응용프로그램에서 비즈니스 로직과 프리젠테이션 로직을 분리 할 수 있는 방법을 제공함과 아울러 표준화된 디렉토리 서비스와 보안 서비스를 제공할 수 있으며, 모듈로 작성되어 있기 때문에 재활용 할 수 있는 장점이 있다. 다음 그림 2-2는 RPC를 사용하는 경우의 클라이언트/서버 구조이다.

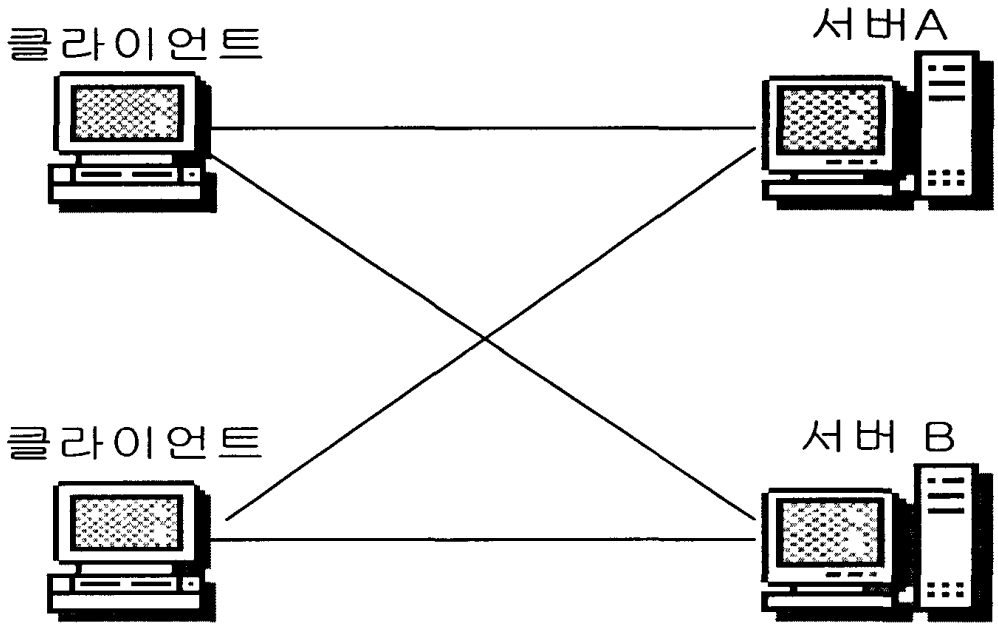

그림 2.2 RPC 클라이언트/서버 구조

그러나, RPC는 주로 C언어를 이용해서 작성해야 하고, 프로그램 작성이 어려우며, 디렉토리 서비스와 보안 서비스를 위해서 프로그램을 직접 작성해야 한다는 문제점을 지니고 있다. 무엇보다 가장 큰 문제는 객체지향 프로그래밍을 지원하지 못한다는 점이다.

2.4 Common Object Request Broker Architecture(CORBA)

CORBA는 전 세계에서 전 세계에서 700개이상의 회사가 참여하고 있는 OMG(Obiect Management Group)라는 컨소시움에 의해 만들어진, 공동 이용 가능한 분산 객체들을 만들고 사용하는 산업체 표준으로 제시된 것이다. 다음 그림은 CORBA는 시스템의 대략적 구조이다.

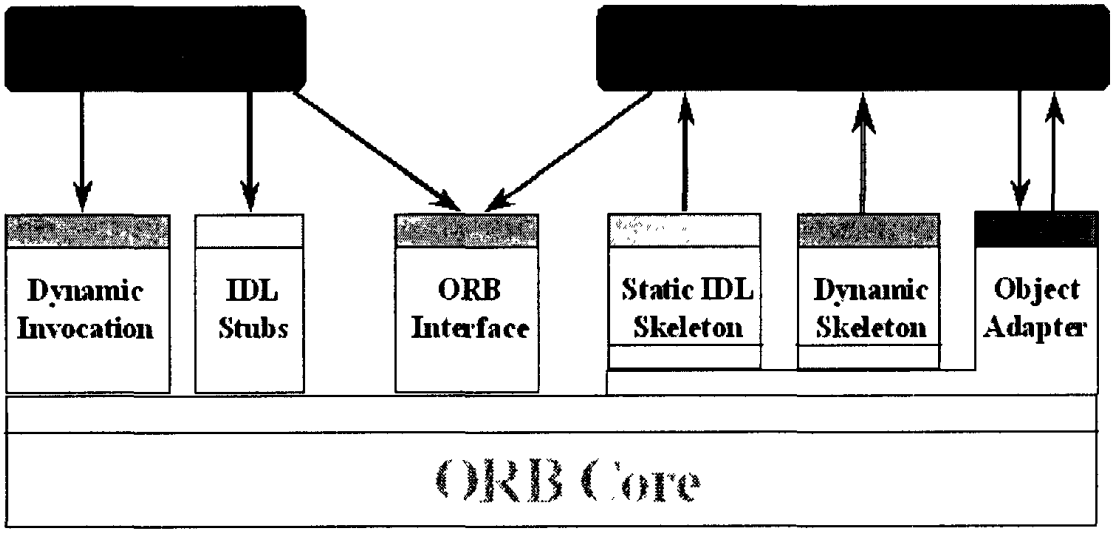

그림 2.3 ORB의 대략적 구조

OMA의 중심에는 ORB가 위치하며, 이것의 구조와 인터페이스는 CORBA의 핵심부를 구성한다. 이것은 어떤 객체가 로컬객체 및 원격객체와 명확한 방식으로 요구와 응답을 주고받을 수 있도록 해주는 일종의 객체 버스이다. ORB는 이 시스템에서 실제 구현을 담당하며 자질구레한 세부사항이 사용자로부터 가려지도록 해준다. 따라서, 사용자는 서버의 참조자와 서버가 지원해 주는 인터페이스만을 알면 된다. ORB의 상위 계층에는 CORBA 서비스가 위치한다. 여기에는 에플리케이션의 다양한 속성을 알아낼 수 있도록 해주는 네이밍 서비스(Naming Service), 영속성(persistence)서비스, 트랜잭션(transaction) 서비스, 보안 서비스 등의 다양한 객체 서비스가 존재한다. 이

서비스들은 많은 도메인들 사이에서 공통적으로 사용되는 것들이며, 현재로는 모두 16개의 서비스가 이곳에 정의 되어 있다. 그 다음은 CORBA 퍼실러티이다. 이것은 복잡한 비즈니스 객체를 구축하는데 사용될 수 있는 에플리케이션 프레임워크의 집합이다. 이것들은 수평 퍼실러티(Horizontal Facilities)를 구성하는 일반적인 프레임워크(common framework)와 수직 퍼실러티(Vertical Facilities)를 구성하는 도메인에 특화된 프레임워크(domain specific framework)로 구분된다.

CORBA 장점 중 대표적인 것은 개방적이고(openness) 상호운용성이 보장되는 시스템을 구축할 수 있다는 점이다. 또 다른 이점은 기존의 시스템과 협력하여 작업하는 새로운 시스템들을 개발할 수 있다. 기존의 에플리케이션과 새로운 에플리케이션이 객체 버스를 통해서 통신하며 동작하는 혼합된 시스템을 구축할 수 있다.

그러나, 일반적으로 Corba 프로그램은 구현이 어렵고, 유지보수에도 제반적 문제가 있는 걸로 알려져 있다.

2.5 Remote Method Invocation(RMI)

RMI를 사용하면 로컬 객체에 있는 메소드를 호출하는 것처럼, 네트워크상의 다른 가상 머신에 존재하는 객체들의 메소드를 호출할 수 있다. 호스트 컴퓨터가 TCP/IP를 사용해서 통신 할 수 있다면, 스트림형식이나 소켓을 사용하지 않고도 네트워크 기반의 애플리케이션을 개발 할 수 있다.

RMI를 사용하는 목적은 그림2.4에서 보는 바와 같이 객체가 어느 곳에 존재하는가에 관계없이, 객체의 메소드를 사용하여 통신 할 수 있도록 하는 것이다. 객체들이 서버에 있던지, 클라이언트가 수행되는 로컬 컴퓨터에 있던지 간에, 그것들이 동일한 런타임 환경에서 수행되는 것처럼 해주는 것을 의미한다. 프로그래머의 관점에서 보면 분산되어 있는 에플리케이션들의 네트워크에 관련된 세부사항들이 가려지게 되어 프로그래밍이 매우 간결해진다.

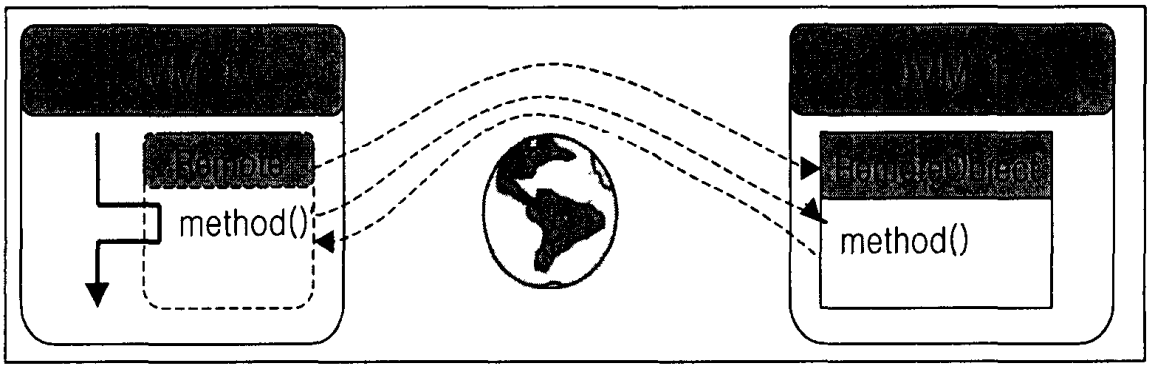

그림 2.4 Remote Method Invocation(RMI)의 개요

RMI에 사용되는 클래스는 그림 2.5에 나와있듯이 네트워크를 통해 접근할 메소드를 정의해 주는 인터페이스를 구현해야 하며, 덧붙여서 자신인 로컬 영역에서 작업하는데 필요한 인터페이스도 구현해야 한다. 그 다음에는 JDK에 포함되어 있는 rmic라는 툴을 사용하여 스텁(stub)과 스켈레톤(skeleton)을 만들어야 한다. 스텁클래스는 원격에 있는 객체의 메소드를 호출할 때 필요한 설정과 매개변수들을 네트워크를 통해 전송할 수 있도록 자동으로 변환

시켜 준다. 스켈레톤은 원격의 가상머신에 존재하는 스텁과 한쌍을 이루는 클래스이다. 이 클래스는 네트워크를 통해 요청을 받고, 그것을 실제 객체상의 메소드 호출로 변환해 주는 작업을 담당한다.

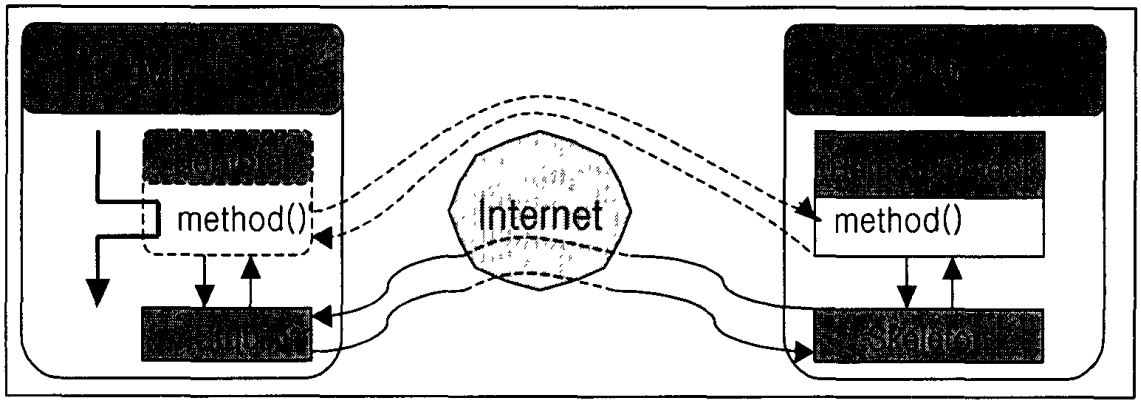

그림 2.5 Remote Method Invocation(RMI)구조

RMI를 사용하는데 마지막으로 필요한 작업은 클라이언트에게 작업이 수행되는 위치를 알려주기 위하여, 원격객체를 네임 서비스(Naming Service)에 등록하는 것이다.(그림 2.6참조) 클라이언트는 네이밍 레지스트리(Naming Registry)에 접속하고, 주어진 이름으로 등록된 서비스의 참조자(Reference)를 요청한다. 그러면 네이밍 레지스트리는 주어진 이름에 해당하는 원격 객체의 참조자를 반환한다.

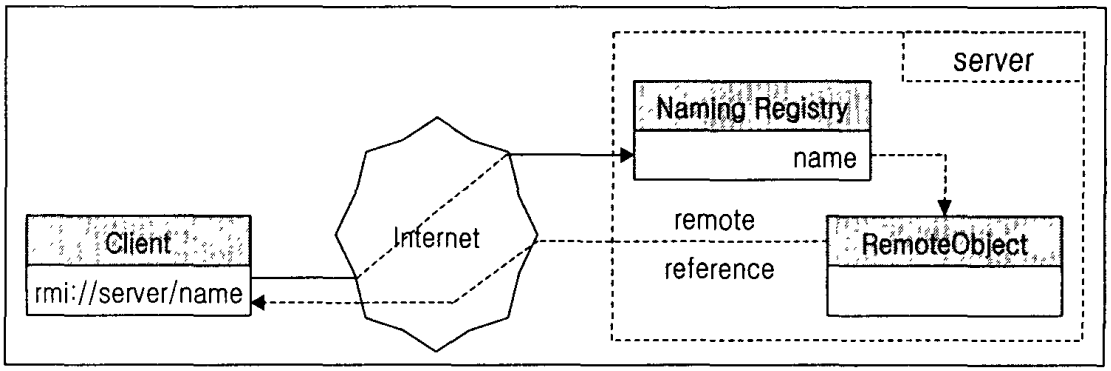

그림 2.6 네이밍 레지스트리(Naming Registry)

2.6 EJB(Enterprise Java Beans, J2EE)

EJB(Enterprise Java Beans)는 어떤회사, 서버 구현, 미들웨어, 통신 프로토콜에 제한되지 않는다. 특징적인 사실은 EJB는 어떤 회사가 여러 다른 방법으로 구현 할 수 있는 명세인 것이다. 만일 어떤 독점적인 확장을 이용하지 않는다면 요구사항이 변경 될 때, 여러분의 EJB를 어떤 서버 구현으로부터 다른 서버 구현으로 옮길 수 있다. 한 마디로 EJB는 재사용 가능한 컴포넌트이다. 다음의 그림2.7을 보면 서로 타입간의 클라이언트와도 작업 할 수 있음을 확인 할 수 있다.

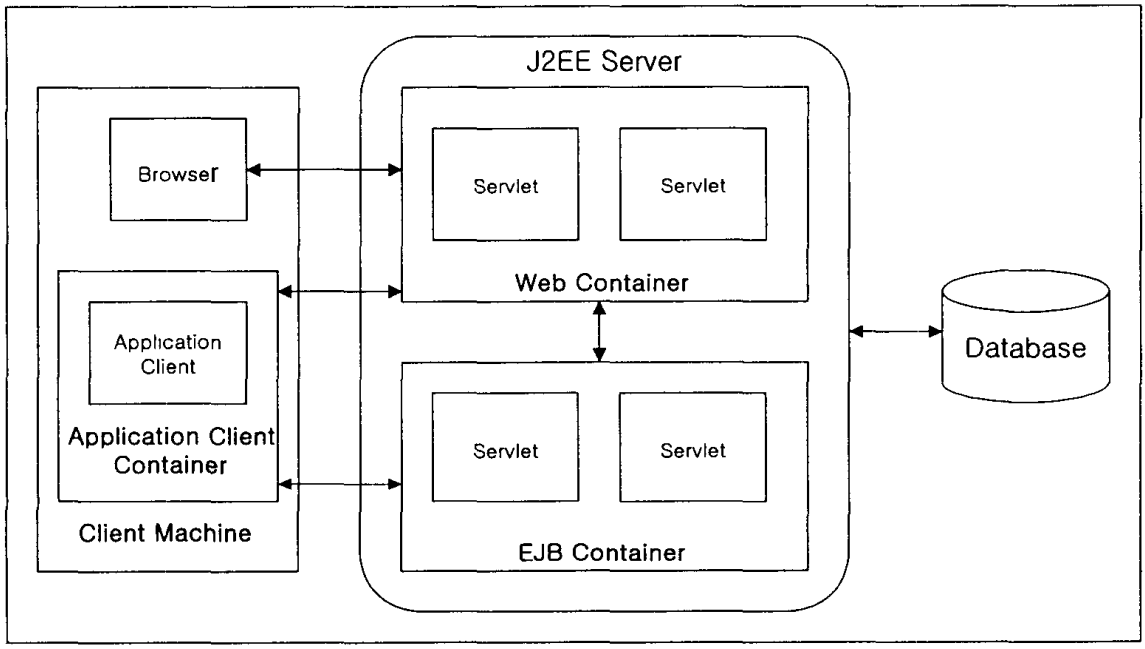

그림 2.7 J2EE 서버와 컨테이너

① 웹클라이언트의 접근을 제공하기 위하여 서블릿과 JSP를 사용 할 수 있다.

② RMI를 사용하는 자바 클라이언트로부터 직접 EJB를 접근 할 수 있다.

③ RMI/IIOP를 지원하는 서버를 접근하기 위하여 CORBA를 사용 할 수 있다.

④ XML을 지원하는 어떤 타입의 클라이언트의 접근을 제공하기 위하여 서블릿을 통하여 XML을 사용 할 수 있다.

EJB는 소규모 작업 그룹 에플리케이션에 대한 개발자의 생산성을 높여주고, 각개의 환경에서 상호 작용하는 웹사이트를 지원하는 모델구조에 도움을 줄 수 있다. 다음의 그림 2.8은 EJB 컨테이너가 제공하는 서비스이다.

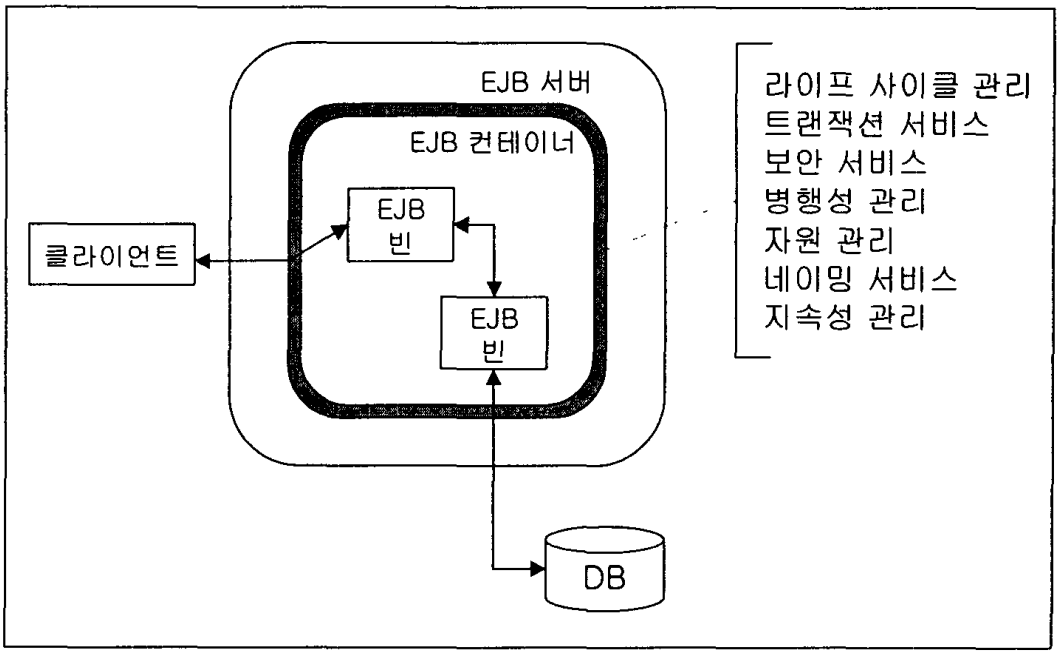

그림 2.8 EJB 컨테이너의 서비스

이러한 기본적 서비스 제공으로 다음과 같은 장점을 얻을 수 있다.

① 개발시간 단축

② 복잡성 감소

③ 소프트웨어의 품질과 신뢰성 증가

④ 재사용성 증가

3. 교육시스템 분석 및 설계

3.1 교육시스템의 주요 기능 및 요구사항 분석

본 시스템이 제안하는 교육시스템의 주된 기능은 교수자와 수강생간의 상호작용 및 공동 작업 환경의 구축이다. 이에 따라 시스템이 가져야 할 수강생 및 교수자의 요구 사항이 몇 가지 존재하게 된다. 크게, 이 시스템의 주된 교육방식은 기존의 전수식 하향 교육방식이 아닌 공통의 주제에 대한 상호간(교수자 및 수강생)의 토론 방식으로 진행된다. 따라서 교수자는 기본적인 주제를 제시함과 아울러 좌장의 역할로 수업에 참여하게 된다. 또한 수강생은 수업에 참여함과 동시에 해당 주제에 대한 연구 및 발표를 네트워크 상에서 자유롭게 행사 할 수 있다. 상세한 사항은 다음과 같다.

- 수강생은 로그인 해야 참여 할 수 있다.
- 사용자 환경(GUI)은 크게 칠판(화이트보드), 웹브라우저, 강의파일게재 등으로 구성한다.
- 교수자는 자신의 강의용 웹페이지나 강의 자료를 칠판(보드)에 게시 할 수 있다.
- 등록된 수강생은 게시된 자료에 질문이나 보충 설명을 그대로 추가 게시할 수 있다.
- 교수자는 수강생의 질문에 대한 답변 및 보충설명을 실시간 적으로 할 수 있어야 한다.
- 수강생이 늦게 시스템에 접근해도 현재까지의 강의진행상황을 그대로 확인할 수 있다.
- 등록된 수강생은 강의종료후 강의 파일을 다운로드 받을 수 있다.

3.2 UML(Unified Modeling Language)을 이용한 시스템 분석

UML(Unified Modeling Language)은 1997년 말에 OMG에서 국제표준으로 채택된 분석/설계 단계에서 사용되는 모델링 언어이다. UML을 사용하면 소프트웨어 시스템의 모델링 결과를 표현력이 강한 시각적 심벌로서 나타낼 수 있기에, 시스템 개발에 참여하는 분석가, 개발자, 사용자 등이 함께 이해하고 모델링 결과를 공유 할 수 있게된다. 또한, UML은 시스템을 모델링하는데 적합하며, 기업 정보시스템으로부터 분산처리의 웹기반 애플리케이션과 실시간 내장시스템까지도 범위에 포함한다. UML은 표현력이 매우 풍부한 언어로서, 그러한 시스템들을 개발하고 배치하는데 필요한 모든 관점을 다룬다.

UML은 모델링 언어일 뿐 메소드 언어는 아니다. 메소드는 프로세스에 대한 정의와 각각의 업무들에 대한 지침과, 업무들에 대한 지침과, 업무들간의 순서들을 명시해야 하는 반면, 모델링 언어는 표기법(또는 다이어그램)들만을 제시하는 것이다. 따라서 UML은 소프트웨어 개발에 사용하기 위한 여러 다이어그램들을 정의하고 있으며, 또 다이어그램들의 의미들에 대해 정의하고 있다.

UML은 여러 가지 다이어그램들을 제시함으로써 소프트웨어 개발과정의 산출물들을 시각적으로 제시하고, 위에서 언급했다시피 분석가, 개발자, 사용자 개개간의 의사소통을 원활하게 할 수 있다. 결국, 개발프로젝트를 성공적으로 수행할 가능성이 높아지게 되며, 프로젝트 초기에 사용자의 요구사항을 제대로 도출할 수 있게 되어, 사용자가 진정으로 원하는 시스템을 개발 할 수 있게 된다. 본 논문에서는 Use Case 모델링과 Class Diagram만을 구현하였음을 알린다.

3.2.1 Use Case 모델링(사용자 관점에서의 시스템 분석)

Use Case란 사용자의 입장에서 바라본 시스템의 특성을 설명하는 것이다. 즉, 사용자들의 시스템에 대한 요구사항 들을 정리해 놓은 것이라 할 수 있다. 교육시스템의 사용자는 크게 두 부류로 나눌 수 있다. 첫 부류는 강의를 진행하는 교수자이고, 두 번째 강의에 참여하는 수강생이다.

교수자는 교육시스템에 로그인해서 강좌를 개설할 수 있어야 한다. 또한 강좌에 참가한 학생들에게 채팅형식을 통해 메시지를 보내고 화이트보드(강의용 칠판)를 이용해 강의를 진행할 뿐만 아니라, 질의에 대한 추가설명도 가능해야 한다. 또한 강의자료를 화이트보드에 디스플레이 할뿐만 아니라, 디스플레이한 자료가 시스템에 로그인한 학생들 모두에게 동일하게 전송되야 하며 그 자료를 통해 수업이 가능하도록 해야 한다.

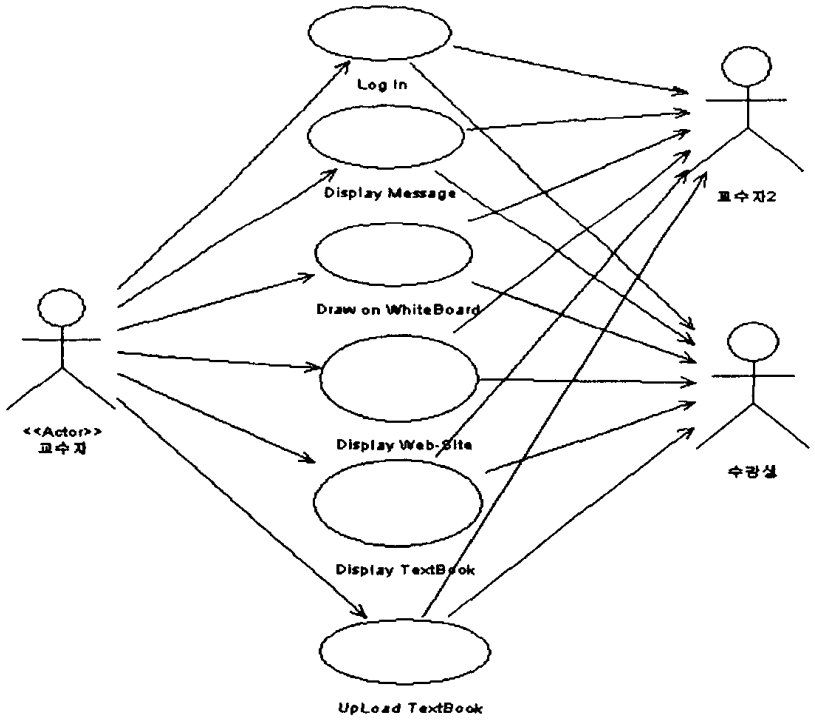

그림 3.1 교육시스템의 Use case Diagram1 (교수자 기준)

수강생 중심의 Use Case 모델도 교수자와 비슷하다고 할 수 있다. 수강생도 마찬가지로 교육시스템에 로그인해서, 강의에 참여 할 수 있고, 수강생과 교수간의 질의, 응답도 채팅의 형식으로 의사교류가 이루어지며, 수강생 상호간의 토론도 마찬가지이다. 실제로 이 교육시스템의 경우, 교수자와 수강생 상호간의 관계는 큰 의미가 없다. 수강생 또한 자신의 준비자료를 화이트보드에 게시할 수 있으며, 자신도 교수자와 동일한 방식으로 강의를 진행 할 수 있다. 이러한 방식을 사용함으로서, 실제로 교수자 1인의 하향식 강의 형태를 벗어나 참여형 강의형태를 유도하며 아울러 토론 방식이 가능하므로 강의의 질적 상승을 유도 할 수 있는 점이 가장 큰 장점이다.

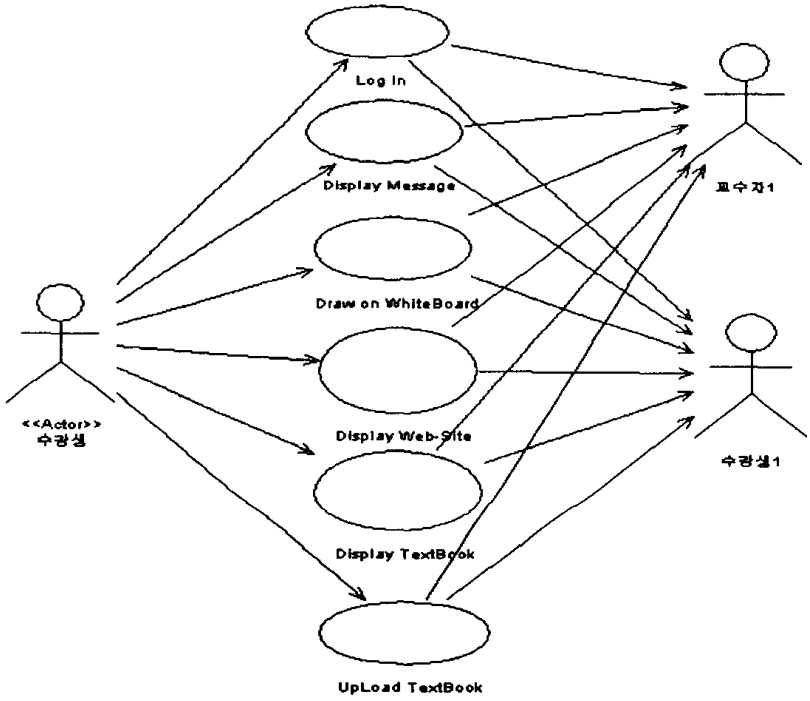

그림 3.2 교육시스템의 Use case Diagram2 (수강생 기준)

Actor	설명
수강생	교육시스템에 로그인하여 학습을 수강하는자
교수자	교육시스템에서 강의를 담당하는 자

표 3.1 각각의 ACTOR의 역할

3.2.2 Class Diagram

Class Diagram은 시스템을 이루는 클래스들과 각 클래스들간의 관계를 나타낸 것이다. 본 논문에서 제안한 교육시스템의 객체는 교수자와 수강생 상호간의 의사소통을 위한 메시지 전송 객체(채팅객체)와 실제적인 강의를 진행하면서 강의자료게시가 가능한 강의보드로 구성된다.

채팅 객체는 위에서 언급했다시피 상호간 의사소통을 위한 것으로, 일반적인 채팅구조와 대동소이하므로 생략토록 하겠다.

강의보드 객체는 크게 GUI 환경을 제공하는 부분과 통신을 담당하는 부분으로 나뉘어 구성된다. 본 논문의 강의보드의 기본적인 설계 구성은 강결합(tightly coupled) 형태가 아닌 약결합(loosely coupled) 형태이다. 다시 말하자면, 서로 다른 클래스들이 강하게 엮여 있지 않다는 점이다. 이러한 방식의 장점은, 원래의 GUI Application을 재구성하지 않아도, 개개의 통신 방식에 큰 영향을 주지 않는 다는 점이다. 또한, 본 논문의 중요한 기능 중 하나인 개개의 접속자 간의 공동 작업환경 구축에도 큰 이점으로 작용하게 된다.(그림 3.2, 그림 3.3참조)

강의보드 객체는 위에서 언급했다시피 실제적 강의가 이루어지는 공간이다. 특징적인 것은 강의보드는 전체 에플리케이션의 네트워킹에 대해서는 전혀 알지 못하며, 관여하지도 않는다는 점이다. 그 대신, 클라이언트는 분산 리스트 자료구조에 기반하고 있다. 분산 리스트 자료구조(Distributed List Datastructure)는 벡터와 비슷한 자료구조이며 표준 API로 조작 할 수 있다. 접속된 클라이언트들 중 누군가가 변경을 가하면 이것이 중앙집중식 데이터 서버를 통해 모든 클라이언트에게 명확히 전달된다. 이 방법은 다시 설명하도록 하겠다.

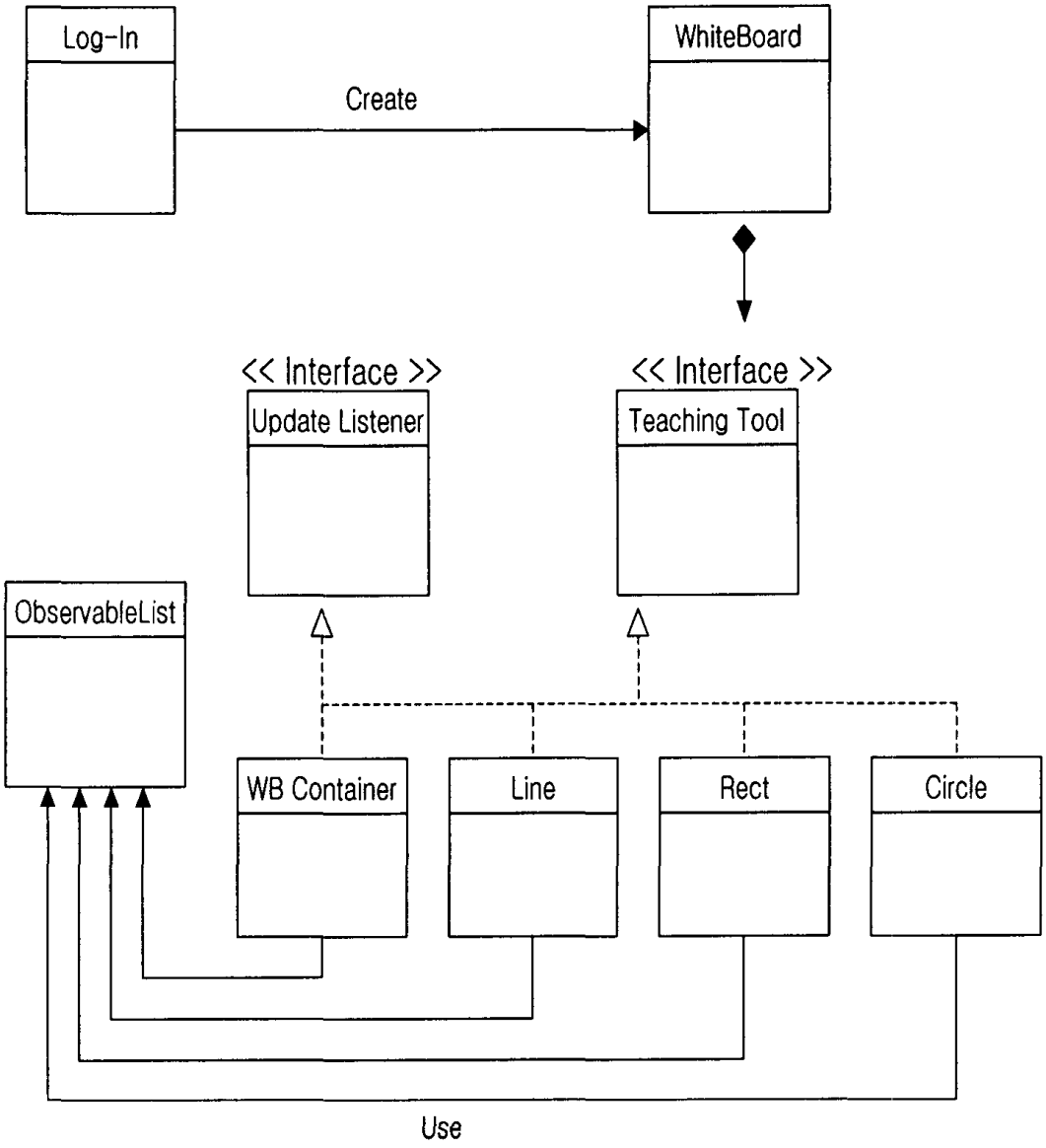

그림 3.3 강의보드객체 GUI class Diagram

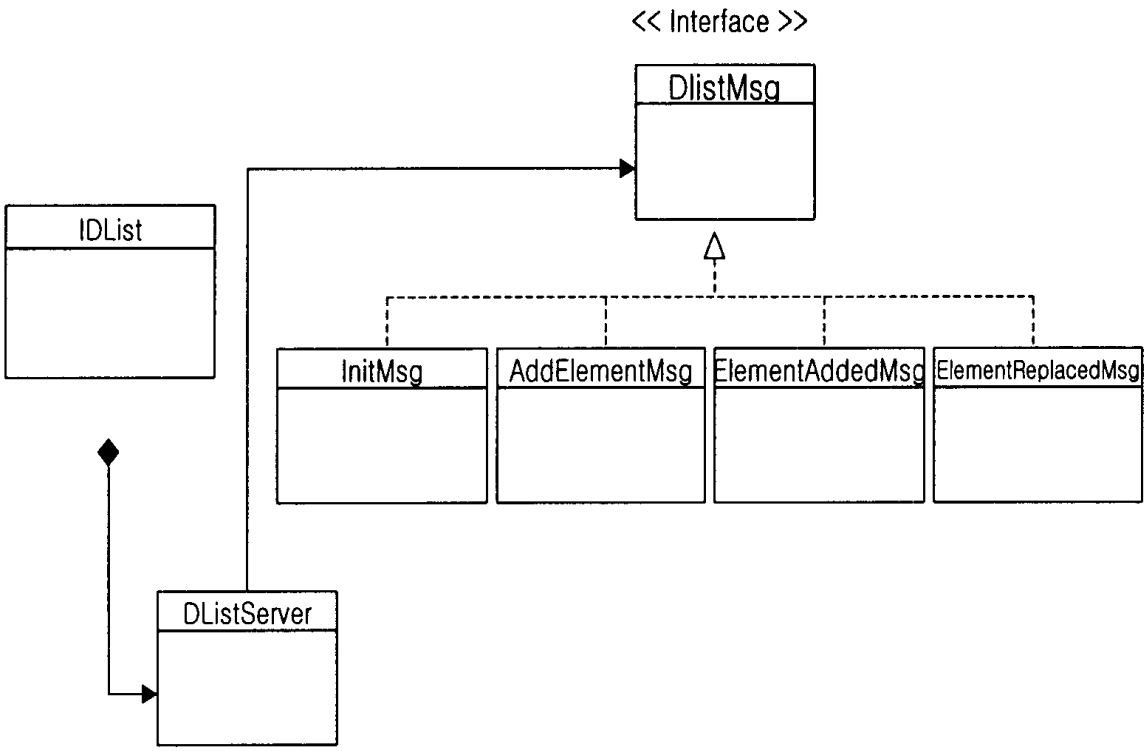

그림3.4 강의보드객체 통신 class Diagram

본 논문에서 제안하는 강의시스템의 화이트보드 프레임워크(Framework)는 분산 리스트 자료구조를 기반으로 만들어진다.(그림 3.4) 분산 리스트는 기본적으로 요소(element)의 리스트이다. 이는 벡터(Vector)와 매우 유사하다. 리스트 내부에서는 인덱스 대신에 내용(contents)에 대하여 관련지어지는 식별자(identifier)를 사용하여 요소를 조작한다. 이 식별자를 각 클라이언트 안에다 저장해두기 때문에, 또 다른 사용자가 리스트의 요소를 변경하고 싶다면 식별자를 통해서만 가능하고, 변경된 식별자가 모든 클라이언트에 동일하게 전송되게 된다.(그림 3.5 참조)

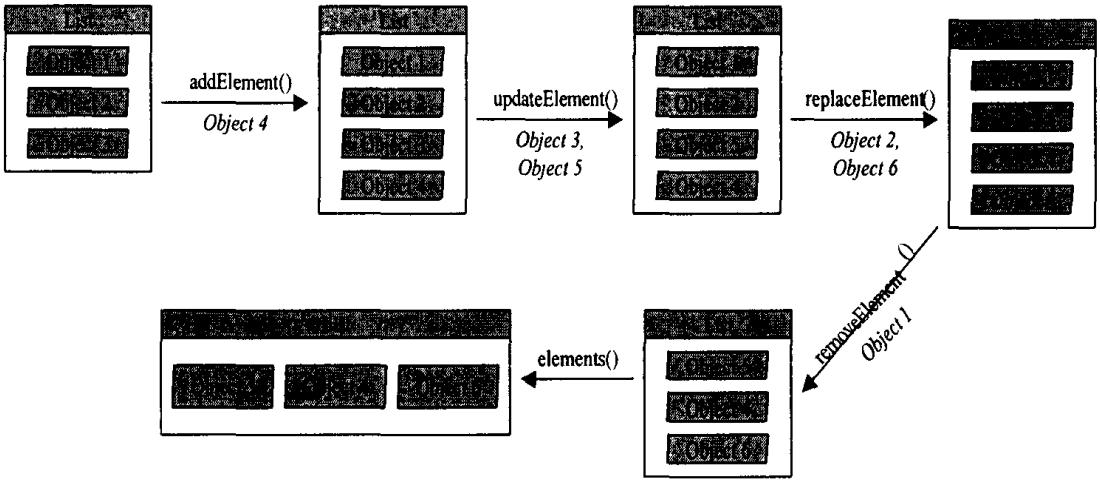

그림 3.5 리스트 자료구조의 개요

이 강의보드의 장점은, 클라이언트 구현 코드를 전혀 고치지 않아도 하부의 네트워크를 쉽게 바꿀 수 있다는 점이다. 이에 따라 본 논문에서는 서론에서 언급했다시피, 클라이언트/서버 방식(Client/Server)과 RMI, EJB 등의 서로 다른 환경에서 이점을 증명하겠다.

4. 교육시스템 구현

이 장에서는 3장의 설계모델을 기반으로 실질적인 시스템의 구현에 대해 언급하고 나아가 각각의 클라이언트/ 서버 환경 및 RMI, EJB 개발 환경 상에서의 구현을 보이겠다.

4.1 교육시스템의 구현

본 논문에서 제안한 교육시스템은 분산환경에서 효율적 개발환경을 제공하는 자바(Java) 및 RMI, EJB로 구현했다. 위에서 언급한 각 개발 언어는 Windows 환경이나 Unix 환경등 다양한 플랫폼 환경에서도 실행이 가능하다. 또한 UML을 사용해 객체지향적으로 설계, 구현하여 프로그램의 확장성 및 재사용성을 높였음을 밝힌다.

일단 사용자가 교육시스템에 로그인 한 후 참여하면 그림 4.2와 같이 강의용 화이트보드가 나타난다.

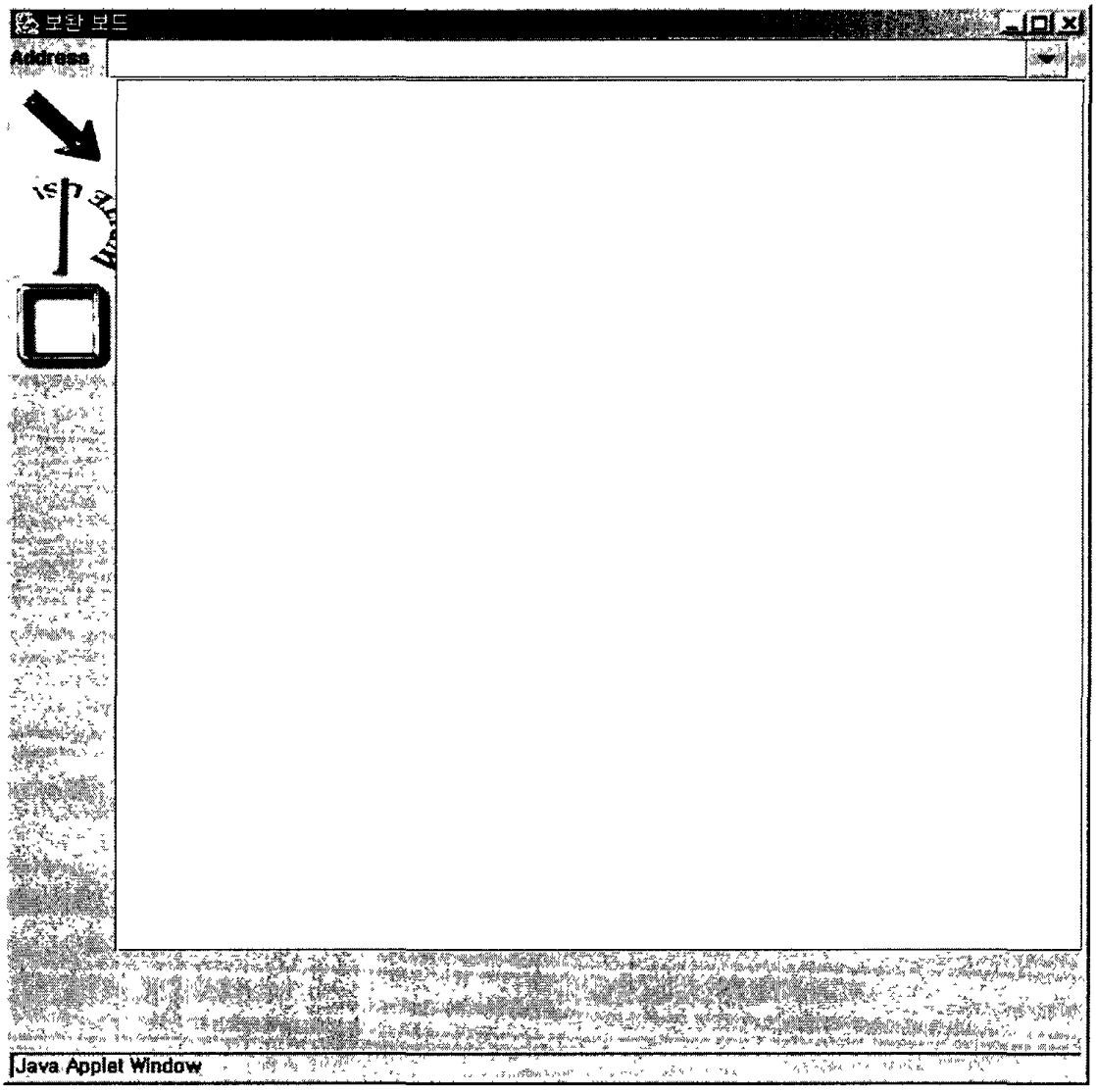

그림 4.1 강의보드 초기 형태

사용자가 강의에 참가하여 그림 4.2와 같이 사용자 인터페이스 환경이 디스플레이 된 후에는 모든 작업이 이 화이트 보드에서 이뤄지게 된다. 또한 다른 사용자들과 채팅을 통해 메시지를 서로 교환하게 된다.

교육진행은 교수자의 강의자료를 게재 할 수 있고, 특정 사이트로 접근하여 진행 할 수 도 있다.

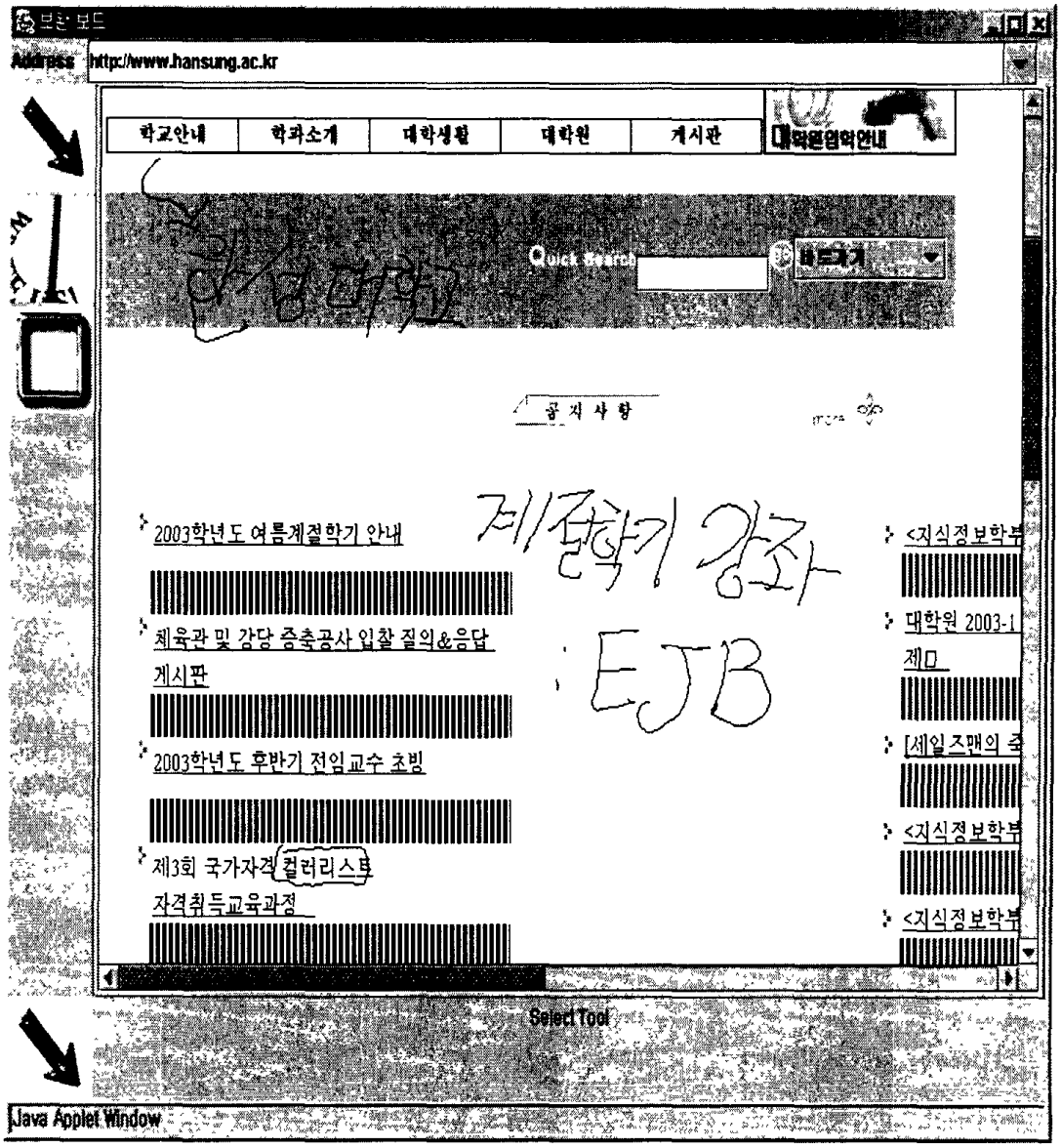

그림 4.2 강의 게재 및 진행 상황(교수자 중심)

자세한 세부적 구현 사항 및 핵심 코드설명은 다음 장에서 설명하겠다

4.2 교육시스템 에플리케이션 GUI환경 구축 사항

① WhiteBoard - 메인 클래스이며 강의 보드의 모든 컴포넌트들을 포함하고 있는 독립실행적 클래스이며 swing을 이용한 JFrame 형태로 구성하였다.
② TeachingTool - 강의 보드의 각종 그리기 도구에 접근하도록 구성된 클래스이며 인터페이스로 작성한다.
③ Element - 그려진 모든 요소들을 의미하며, 강의교재의 교재도 이에 포함된다.
④ WBContainer -경중량 콘테이너로 강의보드에 그려지는 모든 요소들을 디스플레이한다.
⑤ ObservableList - 강의보드에서 그려지는 모든 요소들의 집합을 묶음화한 것으로서, 벡터(Vector)와 매우 유사하며, 리스트에 존재하는 모든 클라이언트에게 일어나는 모든 상황을 통지해주는 역할을 한다. 자세한 메소드는 다음과 같다.

```
//import java.util.*;
//public class ObservableList {
// public Enumeration elements ()..
//  public abstract void addElement (Object element)...
//  public abstract void replaceElementAtEnd (Object oldElement,
//Object newElement)..
//  public abstract void addUpdateListener (UpdateListener l)..
//  public abstract void removeUpdateListener (UpdateListener l)..
//  public void fireUpdate ()..
}
```

Observable 리스트 클래스에는 이 리스트 자료구조의 요소를 액세스 할 수 있는 여러 가지 메소드가 선언되어 있다. 이 클래스에 추가되는 모든 요소는 직렬화 가능 객체인 Serializable 타입이라는 점이 매우 중요하다. 이점 때문에 객체 스트림이나 RMI를 사용하여 리스트 자료구조의 요소를 내보낼 수

있는 것이다. 세부 메소드의 설명은 다음과 같다.

public Enumeration elements ()..

이 메소드는 현재의 리스트 자료구조에 대한 Enumeration 객체(순차적으로 쌓인 객체)를 얻어낸다.

public abstract void addElement (Object element)...

이 메소드는 리스트 요소 element를 이 자료구조의 끝에 추가한다. 단, 분산 환경에 맞추어져 있는 내부적인 구현 방식 때문에, 이 메소드가 호출된다고 해서 곧바로 요소가 추가된다고 생각해서는 안 된다. 그 이유는 네트워크의 속도가 느릴 경우도 고려해야 하기 때문이다. 그 대신, UpdateListener 콜백 메커니즘을 이용하여 정말 요소가 추가되었을 때를 호출측에게 알려주도록 하였다.

public abstract void replaceElementAtEnd (Object oldElement, Object newElement)..

이 메소드는 oldElement를 없애고, 그 대신 newElement를 리스트의 끝에 추가한다. 물론 oldElement가 있지 않을 때는 자료구조의 변경이 일어나지 않는다.

public abstract void addUpdateListener (UpdateListener l)..

이 메소드는 현재의 리스트 자료구조가 바뀌었을 때 콜백에 쓰이는 UpdateListener 객체인 listener를 추가한다.

public abstract void removeUpdateListener (UpdateListener l)..

이 메소드는 자료구조가 바뀌었을 때를 알려 온 listener를 삭제한다.

⑥ UpdateListener 인터페이스

이 인터페이스는 DistributeList의 변경을 알리는 객체의 골격을 나타낸다.

⑦ UpdateEvent 클래스

DistributeList에 대한 변경을 알리는 이벤트를 나타내는 인터페이스이다. 현재는 특별한 정보를 담고 있지는 않지만, 좀더 복잡하게 구현하면 주어진 자료구조에 일어난 변경이 어떤 타입인지까지 담게 할 수도 있다.

⑧ ID 클래스

객체의 식별자를 나타내기 위해 만든 클래스이다. 이 식별자는 int 타입으로 저장된다.

⑨ IDList 클래스

IDList 클래스는 접속된 모든 클라이언트의 변동 사항을 직렬화하여 전송하는 역할을 한다. 위의 ObservableList가 클라이언트의 접속상황과 변동상황을 체크한다면, IDList 클래스는 실제적인 강의 요소의 변동상황 및 객체 전송을 하는 역할을 한다. 특징적인 사실은 자바언어의 clone()메소드를 사용한 점인데, 이는 얕은 복사(shallow copy)를 통해 객체의 직접적인 내용에 접근함이 아닌, 피상적인 참조만을 전송을 위함이다.

```
//import java.io.*;
//import java.util.*;
//public class IDList implements Serializable, Cloneable {
//public synchronized Object addElement (Object element)..
//public synchronized void addElementWithID (Object id, Object
//element)..
//public synchronized Object replaceElement (Object oldID, Object
//element)..
//public synchronized boolean replaceElementWithID (Object oldID,
//Object id, Object element)..
//public synchronized Object getIDOfElement (Object element)..
//public synchronized Enumeration elements ()..
//public synchronized Object clone ()..
//public int getUpdateCount ()..
}
```

4.3 Socket, RMI, EJB환경상에서의 구현

4.3.1 Socket을 기반으로 구현한 강의 보드

본 논문에서 구현한 강의 시스템의 Socket 기반 구조는 기본적으로 일반적인 채팅 시스템의 클라이언트/서버 모델을 그대로 따르고 있다. 클라이언트 측은 위에서 본 GUI 구성에서도 보았듯이 분산 인터페이스화해서 구현하였으나 서버측은 중앙집중식으로 구현하였다.

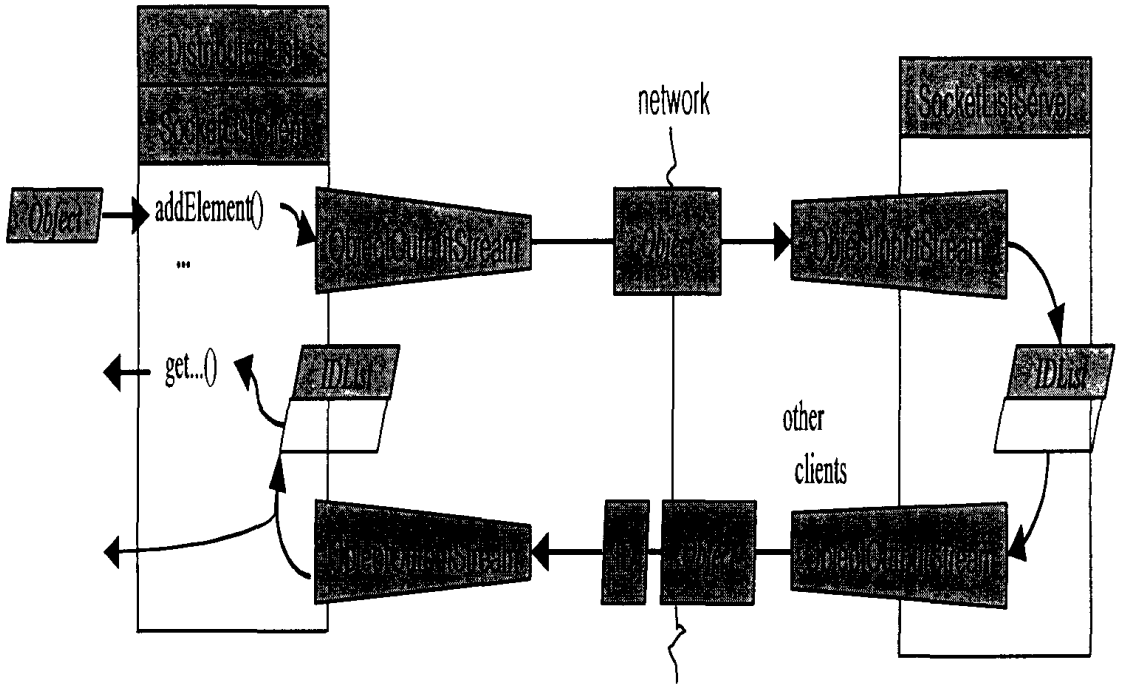

그림 4.3 Socket/Client 기반의 분산 리스트 구조

DistributeListClient 클래스로부터의 메시지는 서버로 전송될 때 IDListMsg 객체로 포장된다. 서버는 이 객체를 받아 포함한 정보를 처리한 다음, 응답 메시지를 IDListMsg 객체로 포장하여 다시 보낸다. 이 구조의 프로토콜은 표4.1과 같다.

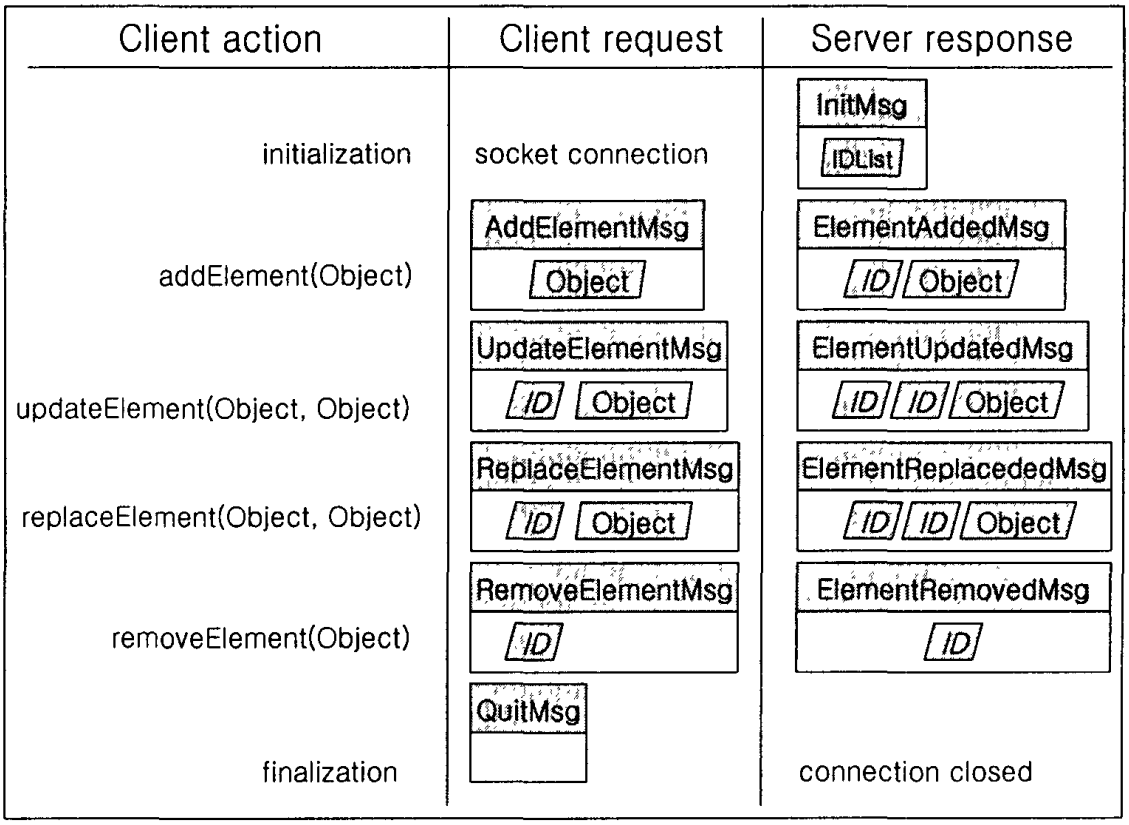

표 4.1 socket 기반의 분산 리스트 프로토콜

4.3.1.1 DistributeListServer 클래스

DistributeListServer 클래스는 소켓기반의 분산 리스트 구조에서 서버 측을 맡는다. 전체 리스트 상태의 주사본(master copy)이 바로 이 서버에 저장된다. 앞서 말했듯이 이 복사본은 얕은 복사(shallow copy)임을 밝힌다.
이 서버는 일반적인 채팅서버의 기본 형태를 따르고 있다. 주 쓰레드는 ServerSocket을 열고 클라이언트 연결을 기다린다. 클라이언트가 자신으로 연결해 들어오면, 이 클라이언트의 모든 요청을 처리하는 핸들러 객체를 새로 만들어 낸다. 우선, 핸들러는 먼저 들어온 리스트의 주 사본을 보낸 후에, 루프를 돌면서 클라이언트의 요청을 읽고 처리해 간다.
주요한 기능을 코드화하면 다음과 같다.

```
//import java.io.*;
//import java.net.*;
//import java.util.*;

//public class DListServer implements Runnable{
// public DListServer (Socket socket)..
// public void run ()..
// public void handle()..
// public void processMsgs (ObjectInputStream objectIn) throws IOException, ClassNotFoundException..
// static void addStream (ObjectOutputStream out)..
// static void removeStream (ObjectOutputStream out)..
// static void broadcastMsg (DListMsg msg)..
// public static void main (String[] args) throws IOException..
}
```

4.3.1.2 DistributeListClient 클래스

DistributeListClient는 중앙 서버인 DistributeListServer에서 관리되는 IDList 객체의 상태를 반영하는 자체 IDList 객체를 가지고 있다. 클라이언트가 DistributeListClient에 변경을 가하면, 이것이 중앙 서버에 대한 요청으로 전송된다. 서버는 연결된 모든 클라이언트에게 변경 메시지를 브로드캐스트(broadcast)한다. DistributeListClient는 이 변경 메시지를 받아 자체의 IDList 객체의 상태를 바꾼 후에, 등록된 모든 리스너(Listener)에게 알린다. 리스너들은 표준 DistributeList 인터페이스를 통해 DistributeListClient 객체를 알아 낼 수 있으며, 자체의 IDList 객체로부터 바뀐 상세 정보를 얻어내게 된다.

주요 기능을 코드화하면 다음과 같다.

```
//import java.io.*;
//import java.net.*;
//import java.util.*;

public class DListClient extends ObservableList implements Runnable {
// public DListClient (String host, int port)..
// public synchronized void stop ()..
// public void run ()..
// public void processMsgs ()..
// public Enumeration elements ()..
// public synchronized void addElement (Object element)..
// public synchronized void replaceElementAtEnd (Object oldElement,
Object element)..
}
```

DistributeListClient 클래스는 Distributed 인터페이스와 그에 딸린 모든 메소드를 구현 하였으며, 중앙 서버로부터의 메시지를 대기했다가 처리할 활성 쓰레드가 필요하기 때문에 Runnable 인터페이스도 동시에 구현하였다. 또한,

쓰레딩과 네트워킹을 위하여 start()와 stop() 메소드도 제공된다. start()가 호출되면, 중앙 서버로의 소켓이 만들어지고, 리스너 쓰레드가 만들어지면서 run()메소드를 통하여 서버로부터의 메시지를 기다리게 된다. stop()메소드는 소켓을 닫고 리스너 쓰레드를 중지시킨다.

4.3.2 RMI(Remote Method Invocation)를 기반으로 한 강의보드

RMI에 기초한 강의보드 구현에서는 위에서 구현한 Socket 기반의 강의보드와 비슷한 방식으로 구현하였다. 즉 클라이언트와 서버는 로컬 IDList 객체에 리스트의 상태를 유지하고 있다가, 클라이언트의 리스트가 갱신되면 RMI를 사용하여 중앙 서버에 자신이 변경되었다는 것을 통지한다. 중앙 서버는 이 변경을 받아들여, 다시 RMI를 사용하여 접속된 모든 클라이언트에게 이 변경을 통지한다. 다시 말하자면, 위의 Socket 기반의 구현과 비교해 볼 때, 네트워크 구조의 차이만 존재하는 것이다.

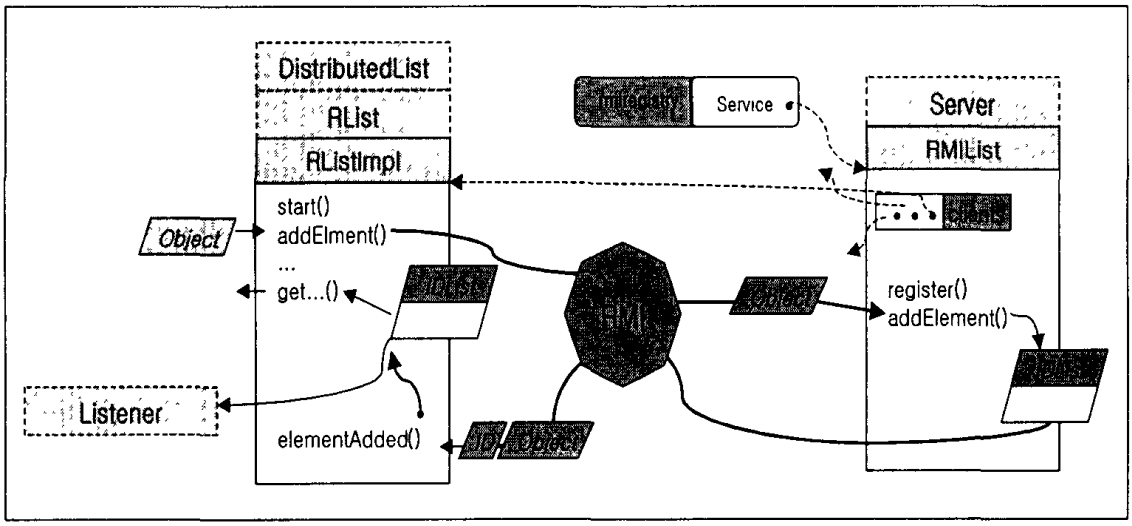

그림 4.4 RMI기반의 네트워크구조

이 시스템은 3개의 클래스로 구성되어 있다. RList는 중앙 서버를 기술하는 원격 인터페이스이다. 클라이언트가 리스트의 변경을 요구할 때는 이 인터페이스를 사용하여 접근한다. RListImpl은 클라이언트 리스트를 기술하는 원격 인터페이스이다. 서버가 클라이언트에게 리스트가 변경된 것을 통지하는 콜백 호출을 할 때 이 클래스를 사용한다. RMIList는 중앙 서버를 구현하는 클래스이다.

4.3.2.1 RList 클래스

이 인터페이스는 클라이언트가 RMI에 기반한 중앙 서버에 접근하기 위해 사용되는 API들을 기술하고 있다. 주요한 코드는 다음과 같다.

```
//import java.rmi.*;
//import Launcher.*;
//public interface RList extends Remote {
//public static final String REGISTRY_NAME = "RList";
//public IDList getUpdate (int updateCount) throws RemoteException;
//public Object addElement (Object element) throws RemoteException;
//public Object replaceElement (Object id, Object element) throws
RemoteException;
}
```

4.3.2.2 RListImpl 클래스

이 클래스는 RList인터페이스를 구현한 것으로 RMI UnicastRemoteObject를 상속받아 구현하였다. 특히, UnicastRemoteObject는 TCP스트림을 사용하여 객체에 접근하므로 객체 직렬화를 가능하게 해주는 기능이 있으므로 이에 사용하였다.

```
//import java.util.*;
//import java.rmi.*;
//import java.rmi.server.*;
//import java.rmi.registry.*;
//public class RListImpl extends UnicastRemoteObject implements RList
{
// public RListImpl () throws RemoteException..
// public Object addElement ()..
// public Object replaceElement ()..
// public IDList getUpdate ()..
//  public static void main ()..
}
```

4.3.2.3 RMIList 클래스

이 클래스는 분산리스트의 서버측을 구현한 것이다. 이 클래스는 접속된 클라이언트들 사이의 변화 상황을 접수하여 통지하는 역할을 한다. 주요한 사항은 다음과 같다.

```
//import java.net.*;
//import java.rmi.*;
//import java.rmi.registry.*;
//import java.util.*;
//public class RMIList extends ObservableList implements Runnable {
//public RMIList (String host) throws RemoteException,
NotBoundException..
// public synchronized void stop ()..
// public void run ()..
// protected synchronized void update ()..
// public synchronized Enumeration elements ()..
// public synchronized void addElement ()..
// public synchronized void replaceElementAtEnd ()..
}
```

RMI기반의 강의보드 구현은 이전에 구현한 소켓기반에 비교해 볼 때 그다지 큰 장점은 없다. 그러나 소켓기반과 비교해서 가장 큰 장점은 에플리케이션 레벨 프로토콜(application Level Protocol)을 사용하지 않고도 프로그램을 간단하게 구현한 점이다. 오직 메소드 호출이라는 개념만으로 시스템을 구현한 점이 가장 큰 장점이라고 할 수 있겠다. 클라이언트/서버 구조의 시스템 확장성 및 유지 보수 측면에서 매우 바람직하다. 그러나 프로그램의 효율적 측면을 고려하면 RMI에 기반한 에플리케이션은 매 업데이트마다 생성되는 TCP/IP연결로 인해서 매우 성능이 떨어지는 면이 있었다. 즉, 높은 대역폭을 가지는 에플리케이션이나 서로 많은 데이터를 주고 받는 에플리케이션을 개발하기에는 어려움이 많다는 것이 본 논문의 생각이다.

4.3.3 EJB(Enterprise Java Beans)를 기반으로 한 강의보드

본 논문의 EJB용 강의보드는 4..1이나 4.2에서 구현한 Socket형태나 RMI형태의 설계구조와 매우 흡사하다. 특히 GUI클래스와 메소드는 거의 비슷하다. EJB의 빈 형태는 상태있는 세션빈과 상태없는 세션빈, 엔터티빈, 등으로 나눌 수 있는데 본 논문은 상태 없는 세션빈을 기반으로 구현하였다. 구조의 개요는 다음 그림과 같다.

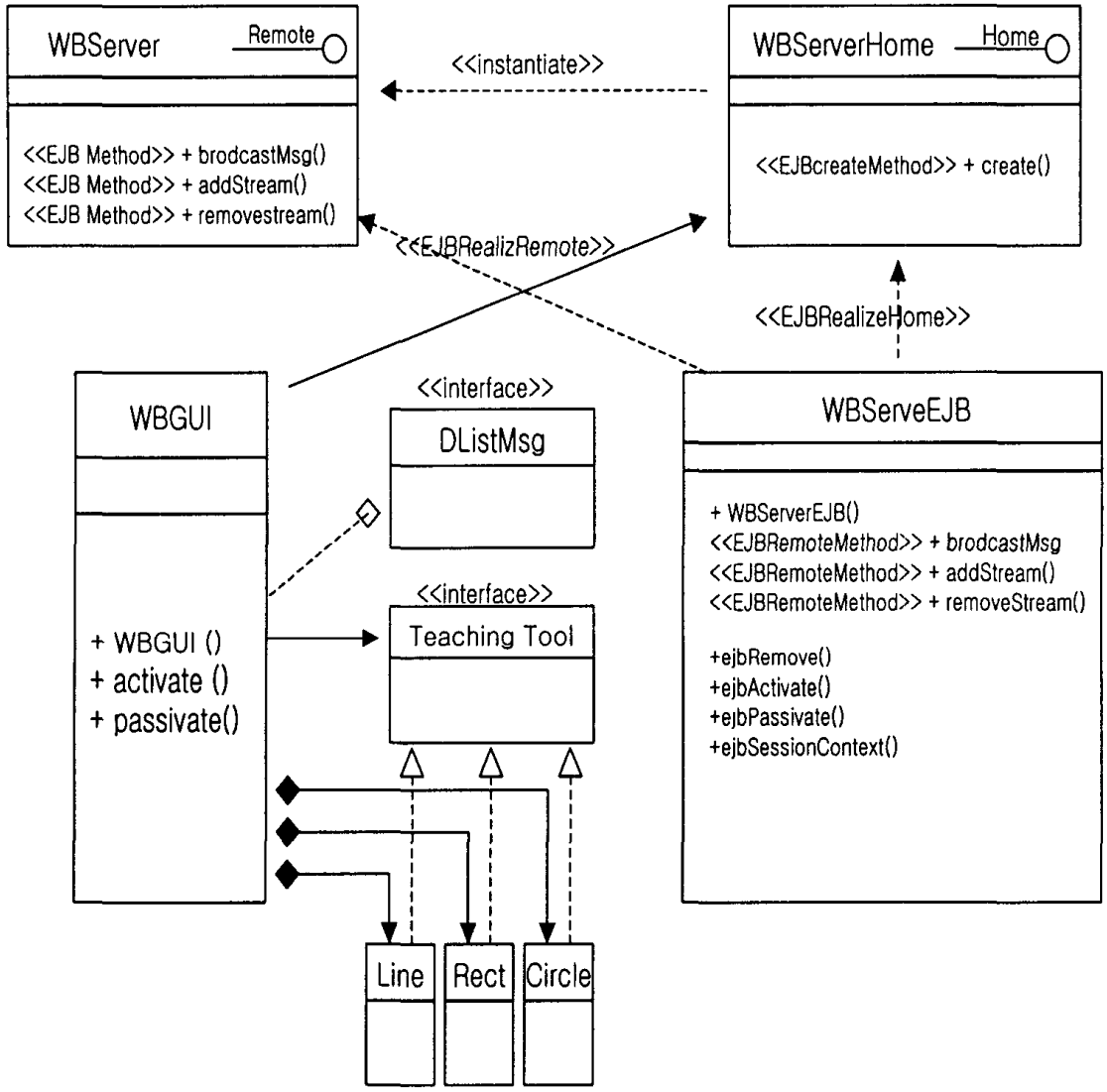

그림 4.5 EJB를 통한 강의보드 설계

이 강의보드의 서버측은 Socket 기반의 에플리케이션과 유사하게 브로드캐스트와 객체자체를 전송하면서 삭제 할 수 있는 메소드도 구현하고 있다. 위에서 언급했듯이 GUI부분은 이미 설계한 바대로 구현을 하였다.

각 객체들은 다음과 같은 순차도에 따라 실행된다. WBGUI는 GUI를 구성하고 active()메소드를 호출한다. WBGUI는 WBServer의 create()메소드를 호출해서 WBServerEJB 객체를 생성한다. 특징적인 사실은 클라이언트는 RMI형태이므로 스텁과 스켈레톤을 생성해야한다.

5. 결론 및 향후 과제

컴퓨터 네트워크가 급속하게 발전함에 따라, 이를 이용한 원격교육에 관한 연구가 활발하게 진행되어 왔다. 컴퓨터 네트워크를 이용한 원격교육은 전통적인 교육의 시간적, 공간적 제약을 극복 할 수 있을 뿐만 아니라 보다 많은 사람들에게 경제적이고 효율적인 교육을 제공한다. 또한 컴퓨터를 이용함으로써 다양한 멀티미디어 등을 통해 좀더 효과적인 교육을 가능케 한다.

이러한 장점들로 인해 원격교육은 컴퓨터 네트워크를 응용한 분야들 중에서도 각광 받는 분야 중의 하나가 되었으며, 지금까지 많은 연구들을 통해 다양한 원격교육 시스템이 제안되었다. 하지만 기존에 제안된 원격교육 시스템들은 대부분 클라이언트/서버 구조를 택함으로서 확장성과 속도 면에서 많은 문제점이 노출되었다.

본 논문에서는 이러한 기존 시스템들의 문제점을 해결하기 위해 분산환경 상에서 구현한 새로운 형태의 원격교육 시스템을 제안하였다. 클라이언트/서버 구조 및 분산환경 상에서 유용하게 쓸 수 있는 자바 RMI, EJB를 기반으로 시스템을 구현하여 각 시스템간의 장단점을 확인해 보았다. 또한 UML을 통해 객체 지향적으로 설계, 구현되었다.

하지만, 본 논문은 교육의 기본적 기능만을 구현하였기 때문에, 실질적인 교육적 효과를 위해서는 추가 보완해야 할 사항이 많은 점이 노출되었다.

첫째, 시청각 교재를 보다 폭 넓게 사용할 수 있도록 기능을 보완해야 한다. 본 논문의 주된 메시지 전달 방식은 일반 텍스트 메시지를 전송하는 것에 불과한 채팅 방식이므로 교육의 효과적 전수에는 매우 많은 문제점이 있었다.

둘째, 개개의 네트워킹 환경 상에서 가장 적합한 구현 방식을 찾는 점이 취약하였다. 성능 면이나 효과 면에서는 클라이언트/서버 방식이 가장 좋았지만, RMI나 다른 분산 프로그래밍 환경과의 비교를 통해 개개의 장단점을 확인하는 점이 미약하였다.

셋째, 교육과정의 특성에 맞는 교보재의 기능을 추가하지 못한 점이다. 본 논문의 기본적 교육기능 외에 다른 기능을 추가로 사용 할 수 있게끔 한다면 보다 유용한 원격 교육 시스템으로 사용 될 수 있다고 생각한다.

마지막으로 본 논문을 완성하게끔 많은 도움을 주신 지도교수님과 학형 여러분께도 감사를 드린다.

참고문헌

[1] 김일민, 알기 쉬운 중급 자바, 홍릉 출판사 2001

[2] Computer Networking, James F.Kurose, Keith W.Ross, Addisson Wesley 2001

[3] Merlin Hughes외 3인, JAVA NetWorking 2000

[4] Karl Avedal, Java Server Programming J2EE Edition 2001

[5] Hans-Erik Eriksson and Magnus Penker, UML ToolKit Wiley Computer Publishing 2000

[6] Stephen Stelting, Olav Maassen, applied JAVA Patterns. SUN MicroSystems 1999

[7] Jess Garms, Daniel Somefield, Professional JAVA Security Information Publishing Group 2002

[8] Hyun-Chul Kim, Distance Learning On Distributed Object-oriented Virtual Environment 2001 Korea Univ.

[9] Hee-man Lee, Analysis and Design of the Remote Education System based on EJB, 2002 SungKyunKwan Univ.

[10] 황대준 외 6인, "21세기형 첨단학교·가상대학 설립운영에 관한 연구", 교육부 교육정책과제 연구보고서. 1997

ABSTRACT

Design and Implementation Education System on Distributed Environment

YU, Jae-Haw

Computer New Technology
of the Graduated School
Hansung University

advised by Kim, il-min

With Dynamic growth of computer network, Distance Education System is researched in many years. Moreover, Using Computer provides more effective Education with Multimedia.

Due to these benefits, distance education became one of the most promising applications of computer networks, and a variety of distance education systems have been presented so far. However previous systems have problems in scalability and speed because most of them are based on the client/server System.

This thesis, we present new distance education system on distributed environment to solve these problems, and this system has been improved in scalability and speed. Furthermore, user interaction is more effective to professor and students. For extensibility and reusability, we designed and implemented the system according to the UML(Unified Modeling Language) and the object-oriented paradigm.