확장성을 고려한 산업디자인 협력시스템 설계 및 구현

(Design and Implementation of an Industrial-Design Collaborative System to Support Scalability)

양 진 모 † 이 승 룡 †† 전 태 웅 †††
(Jinmo Yang)(Sungyoung Lee)(Taewoong Jeon)

요 약 본 논문에서는 3D 애니메이션, 컴퓨터 게임, 산업디자인 제작과 같은 애플리케이션을 원격지에서 가상공간을 통해 공동작업을 수행 할 수 있는 협력시스템 프레임워크 설계 및 개발에 대한 경험을 기술한다. 개발된 시스템은 클라이언트/서버 구조를 가지며, 플랫폼에 독립적이고, 확장과 이식이 용이한 협력시스템 프레임워크 구축을 목표로 하고있다. 이러한 목표를 달성하기 위하여 서버는 자바로 구현하였고, 작업의 부하를 효과적으로 분산시키고 확장성과 객체관리를 용이하게 하기 위하여 분산형과 중앙집중형을 혼합한 절충형 구조를 채택하였다. 이를 위하여 서버는 작업 부하와 기능을 고려하여 사용자의 협력 준비단계를 담당하는 사용자 관리서버, 협력 작업 세션을 지원하는 세션 관리자 서버, 사용자 관리서버들 간의 연동 방법을 제공하는 정보 서버로 구성하였다. 클라이언트는 사용자의 편의를 위하여 Windows환경에서 Visual C++로 구현하였다. 그리고, 공동 저작 도구로써 단독환경 (stand-alone)에서 가장 많이 사용되는 Kinetic사의 상용 3D Studio Max를 객체공유 방법을 제공할 수 있는 plug-in기술을 사용하여 분산환경에서 작동하도록 확장하였으며, 채팅과 화이트보드 기능도 제공한다.

Abstract This paper describes our experience to design and implementation of a collaborative system framework that allows to develop certain collaborative applications such as 3D animation, computer game, and industrial design. The collaborative system enables users, who located in geographically long distance, to do collaborative work in a single virtual space. The proposed system basically consists of client and server system. The goal of proposed system is to support scalability, portability, and platform independent. In order to achieve these, the server is implemented in Java platform and is adopted to the hybrid architecture which takes the advantages both in centralized and decentralized collaborative system. We construct the server base on its functional characteristics so called User Manager Server (UMS), Session Manager Server (SMS), and Information Server (IS). The UMS manages the users who are taking part in the collaborative operations. The SMS supports the conferencing in the proposed system. The IS provides the connection methods among the UMSs. For user's convenience, we implement the client using Visual C++ in Windows. We also expend the functions of 3D Studio Max to distributed environment by means of the plug-in module, and facilitate the chatting and white board functions as well.

1. 서 론

협력 시스템은 원격지의 많은 사용자들이 가상공간에서 공동작업을 수행할 수 있는 환경을 제공 해주는 시스템으로 인터넷, 멀티미디어, 가상현실 그리고 고성능 마이크로 프로세서 기술의 급속한 발전에 힘입어 구현이 가능하게 되었다. 이러한 협력 시스템은 멀티미디어, 산업디자인, 3D 애니메이션, 만화 및 영화 산업, 제품설계, 의료, 건축, 자동차, 선박, 항공 등 전 산업분야의 제품 생산과정에 응용되어 경제적으로 많은 비용절감 효

† 비 회 원 : 경북대학교 전자계산공학과
isoli@oslab.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자계산공학과 교수
sylee@oslab.kyunghee.ac.kr
††† 종신회원 : 고려대학교 전산학과 교수
jeon@tiger.korea.ac.kr
논문접수 : 2000년 1월 20일
심사완료 : 2000년 7월 3일

과를 가져다 줄 수 있을 뿐만 아니라 원격 의사소통, 원격교육, 전자상거래에도 응용될 수 있다. 그러나 협력 시스템은 공유객체의 일관성, 빠른 반응시간 및 영속성 유지, 동시성 제어, 자원 관리, 보안, 시스템 통합 등 아직도 해결해야 할 많은 기술적 난제를 가지고 있다[1].

일반적으로 협력 시스템은 원활한 협력 작업을 지원하기 위해 다음의 조건을 만족시켜야 한다[2][3]. 첫째, 작업에 참여한 참여자들의 행동을 관찰할 수 있는 방법을 제공하여야 하며 (awareness), 둘째 협력 작업은 분산 환경에서 다수의 참여자가 동일한 공유 데이타의 조작에 의해 진행된다. 이때, 각 참여자는 동일한 공유 데이타에 의한 작업 수행을 보장받아야 하며 이를 위해 동시성 제어 방법이 필요하게 된다 (concurrency control). 셋째, 분산된 환경에서 공유 데이타 및 공유 가상 세계를 통해 참여자들 사이에 연속적인 협력 작업이 수행되기 때문에 공유 데이타 조작을 빠른 시간에 참여자 모두가 관찰 할 수 있도록 하여야 한다 (responsiveness). 이를 위해서는 네트워크 지연이 적어야되고 일관성 유지를 위한 효율적인 동시성 제어 방법이 제시되어야 한다. 넷째, 클라이언트의 수에 관계없이 서버는 일정한 성능을 보장해야 한다 (scalability). 다섯째, 협력 시스템은 사용자 사이의 상호작용을 통해 변화된 공유 데이타를 영속적으로 보존할 수 있는 방법을 제공하여야 한다 (persistency).

협력 시스템 서버구조는 중앙 집중형, 분산형, 절충형(hybrid)으로 분류된다. 중앙 집중형 방식은 중앙 서버에 부하가 집중된다는 단점이 있지만 공유 객체의 일관성 유지하기가 용이하며, 분산형 방식은 부하를 분산시킬 수 있는 반면 공유 객체의 일관성 유지가 어렵다. 절충형 방식은 공유 객체와 공유 객체 관리 정보를 분리하여 집중형과 분산형 방식의 장점을 절충한 방식으로 공유 객체들을 각 시스템으로 분산시키고 관리정보를 중앙 관리 시스템에서 유지한다.

본 논문에서는 개발 주기가 짧은 산업 제품의 3D 디자인 협력 공동작업 어플리케이션에 적용할 수 있는 협력시스템 설계 및 구현에 대한 경험을 기술한다. 본 논문에서 다루는 시스템의 목표는 플랫폼에 독립적이고, 확장과 이식이 용이한 프레임워크를 구축하는데 있다. 이를 위하여 프레임워크 모델은 작업의 부하를 효과적으로 분산시키고 객체관리를 용이하게 할 수 있는 절충형으로 설계하였다. 구현된 시스템은 서버와 클라이언트 구조를 가진다. 서버 시스템은 유지 보수가 용이하고 플랫폼에 독립적이며 이식성이 뛰어난 Java 기술을 사용하였다. 또한, 사용 환경에 따라 시스템 확장을 용이하게 위하여 서버를 기능에 따라 사용자 관리 서버 (User Manager Server: UMS), 세션관리 서버 (Session Manager Server: SMS), 정보관리 서버 (Information Server: IS)로 분류하였다. UMS는 사용자 인증 및 상태 정보 제공, 사용자 검색, 실시간 메시지 전송을 지원한다. SMS는 데이타 베이스 관리, 세션 관리, 3D Studio Max의 3D 객체 관리, 화이트보드의 2D 객체 관리를 담당한다. IS는 서버 인증, UMS들 사이의 정보 교환과 메시지 전송 등을 지원한다. 이와 같이 기능별로 구분된 서버 구조는 부하를 쉽게 분산시킬 수 있을 뿐만 아니라, 필요시 IS 서버에 UMS를 용이하게 첨부하거나 삭제할 수 있어 융통성과 확장성을 지원할 수 있다. 클라이언트 시스템의 경우 향후 협력시스템에 응용 프로그램의 개발과 추가가 용이하도록 클라이언트의 세션과 협력 응용(collaborative application)간에 인터페이스를 고려하여 설계되었다.

본 논문에서 다루는 시스템 사용목적은 단독환경에서 사용되는 디자인 툴인 3D Studio Max의 plug-in 모듈을 개발하여 분산협력 환경에서 산업디자인 응용 작업을 가능하게 한 것이다. 또 다른 목적은 제안된 시스템 통해 실제 협력시스템이 가지고 있는 다양한 문제들을 어떻게 해결할 수 있는지를 보여주기 위함이다. 예를 들면 서버의 부하분산 방법 (중앙형/분산형/절충형의 장단점), 객체공유 방법 (이벤트 처리 또는 록킹을 이용한 동시성제어), Java를 이용하여 이기종 플랫폼에 독립적인 시스템 구현 방법과 서버 모니터링, 클라이언트 및 서버 구축방안 등이 그것이다.

본 논문에서 다루는 시스템의 기능은 크게 응용시스템, 서버 그리고 클라이언트로 나눌 수 있다. 응용 시스템 기능은 3D Studio Max를 이용한 분산 산업디자인 협력 작업이다. 서버 기능으로는 첫째, 일관성 및 영속성 유지를 위한 이벤트처리와 공유데이타 관리 기능 둘째, 사용자 인증, 사용자 상태 정보 제공, 사용자 검색, 실시간 메시지 전송과 같은 사용자 관리 셋째, 산업디자인 프로세서 처리를 위한 데이타 베이스 관리, 세션 관리, 3D Studio Max의 3D 객체 관리, 화이트보드의 2D 객체 관리와 같은 세션 관리 그리고 마지막으로, UMS 인증, UMS들 사이의 정보 교환 또는 메시지 전송 등과 같은 정보관리자 기능이 있다. 클라이언트 기능으로는 서버 접속, 세션 관리, 3D Studio Max 객체공유, 화이트보드, 채팅 등이 있다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고, 3장에서는 본 논문에서 다루는 산업 디자인 프로세서에 대해 기술한다. 4장에서는 협력 시스템 서버

구조를 기술하며, 5장에서 이를 지원하는 협력 시스템 클라이언트 모델을 설명한 후, 6장에서는 실제 구현된 시스템의 사례와 성능평가에 대하여 소개하고, 마지막으로 7장에서 결론을 내린다.

2. 관련 연구

협력 시스템은 [그림 1]과 같이 시스템이 지원하는 상호 작용과 사용자의 지리적인 분산 정도에 따라 분류된다. 상호작용이 발생되는 시각에 따라 비동기화와 동기화로 나누어지고, 지리적인 분산정도에 따라 같은 지역에 함께 배치되는 colocated와 다른 지역에 배치되는 광역이 있다[3].

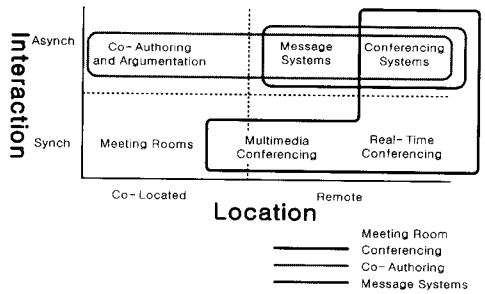

그림 1 협력 작업 분류

Habanero[9]는 Java로 개발된 클라이언트/서버 환경의 협력 시스템 프레임워크로 Java 객체를 인터넷을 통해 공유 할 수 있는 환경을 지원하고, 협력 응용 개발을 위한 API를 제공한다. Java 객체의 공유라는 특징 때문에 인터넷을 통해 플랫폼 독립적인 협력 시스템을 구축할 수 있는 장점을 가지나, 기존의 개인 PC상에서 작동되는 응용 소프트웨어와는 연동이 미약하다는 단점을 가진다.

GroupKit[10]은 Calgary 대학에서 개발한 실시간 응용 개발을 위한 협력시스템 툴킷으로 [그림 2]와 같은 구조를 가진다.

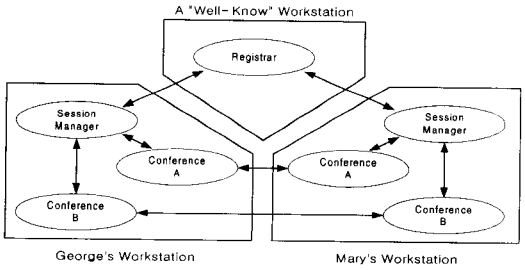

그림 2 GroupKit의 구조

레지스트라(registrar)는 분산된 시스템의 정보를 저장하고, 접속된 각 시스템에서 생성된 컨퍼런스(conference)의 목록 정보와 사용자를 구분할 수 있는 고유 계정을 제공하고, 사용자 정보를 저장한다. 또한, 이를 접속한 시스템에게 보내어 현재 협력 작업이 가능한 사용자 및 개설된 컨퍼런스 정보를 제공한다. 사용자는 레지스트라에 접속하여 사용자 정보를 검색하고 컨퍼런스 정보를 획득하여 협력 작업에 참여한다. 협력 작업은 각 시스템 사이에 peer-to-peer 방식으로 수행되어 각 시스템에게 협력 작업에서 요구되는 기능을 부담하여야 하는 반면, 중앙 서버에 관리 각 시스템의 관리 정보만을 제공하여 타 시스템의 상태를 빠르게 분석 할 수 있고, 협력 작업 시 발생되는 부하가 각 시스템에게 분산된다.

Egret[11]은 Hawaii 대학에 개발된 협력 시스템 프레임워크로 그룹의 공동 작업을 지원하고, 동적이며 확장 가능한 프레임워크 구축을 목표로 개발되었다. 이는 multi- client, multi-server, multi-agent 구조로 되어 있으며 클라이언트/서버 환경의 데이타 베이스 구조를 사용한다. 서버의 부하를 줄이기 위해 서버의 데이타 처리부를 간단히 하고, 클라이언트에게 부하를 분산하였다. 공유 데이타는 하이퍼텍스트(hypertext) 기법을 사용하여 서로 연결되어 있으며, 이를 효과적 처리하기 위해 [그림 3]과 같이 G-table을 정의하였다.

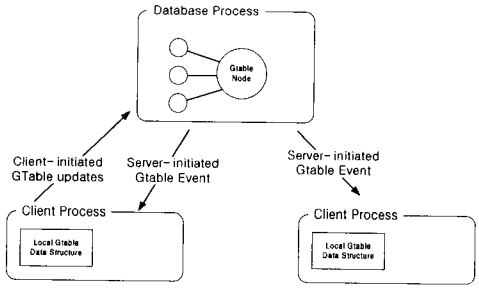

그림 3 Egret의 GTable의 구조

CW-MAN[12]은 효율적인 멀티미디어 공동 저작을 위한 혼합형 구조의 공동 저작 관리 시스템을 제안하였다. 이는 공동 저작 관리 정보를 중앙 관리 시스템에 집중시키고, 공동저작에 사용되는 공유 객체들은 각 저작자 시스템에게 분산시키는 혼합형 구조를 갖는 시스템 모델이지만 아직 구현은 되지 않았다.

[13]에서는 웹에서 플랫폼에 독립적으로 공동 작업을 지원하는 시스템을 개발하였다. 이는 웹 브라우져를 통

해 협력 작업을 수행함으로써 협력 작업에 필요한 부가적인 소프트웨어의 구입 및 설치 과정이 필요 없는 장점이 있으며, 웹을 통에 언제라도 새로운 버전의 소프트웨어를 통한 협력 작업이 가능하다. 하지만, 웹 브라우져에서 제공하는 기능에 의해 협력 작업이 제약받을 수 있으며 동시성 제어 등은 고려되지 않았다.

CoDraw[18]는 공동 드로잉 작업도구의 시스템 구성, 동시성 제어 정책 등에서 유연성을 강화시켰으나, 시스템 응용환경이 주로 드로잉 공동작업에 한정되었다.

[19]에서는 ITU-T의 T.130을 근간으로 공동작업을 위한 회의제어기, 다자간 통신서비스와 공유 애플리케이션을 분리하여 분산객체 공유시스템을 제안하였다.

3. 협력시스템의 산업 디자인 프로세서

앞에서 언급한 바와 같이 본 논문에서 다루는 협력 시스템 응용은 개발 주기가 짧은 산업 제품의 3D 디자인 협력작업이다. 본 장에서는 제안된 협력 시스템을 위한 디자인 프로세서를 간단히 소개한다 (산업디자인 프로세서의 자세한 내용은 본 논문의 범위에 벗어난다).

현재 대기업을 중심으로 한 산업디자인 협력작업은 제품개발을 위해 각 협력 부서들(마케팅, 디자인, 엔지니어링)간의 스케줄에 따른 회의 형식에 기반을 두고 의사결정 및 자료교환을 통해 이루어지고 있다. 제안된 산업 디자인 협력시스템은 마케팅, 디자인, 엔지니어링 부서간의 협력과정에서 시공간적인 제약에 따른 커뮤니케이션의 단절을 보완함으로서 시스템의 실효성을 증진시키는데 목표를 두고 있다. 이를 위해서 협력 부서들간의 회의 및 의사 결정 스케줄을 순차적으로 서버에 데이타베이스화하여서 저장하여 시스템 사용자로 하여금 타 협력 부서 사용자들과 실시간으로 커뮤니케이션을 수행 할 수 있게 한다. 사용되는 툴 들은 채팅, 화이트보드, 3D Studio Max등 이 있다.

[그림 4]는 본 논문에서 다루는 산업 디자인 협력 시스템의 전체 제품 개발 과정을 도식화한 것이다. 제품 개발 과정은 마케팅 부서(Marketing: M), 디자인 부서(Design: D), 엔지니어링 부서(Engineering: E)에 의해 선행 연구, 컨셉 개발, 형상 결정, 디자인 구체화, 상품 개발을 수행하는 5단계로 구성된다. 작업프로세스(work process) 하단의 상자에는 개발 단계의 부서별/부서간 단계별 목표가 정의되어 있으며, 우측의 플로우는 상세 개발 내용 및 상세 협력 구조를 나타낸다. 그리고, 하단의 음영 영역의 상자는 중요 의사 결정 단계와 의사 결정 단계별 협력 과정과 중요 데이타에 대한 설명을 나타낸다.

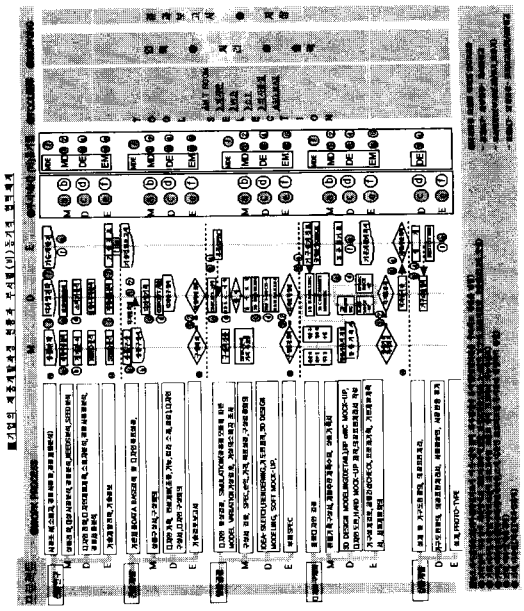

그림 4 산업 디자인 프로세서

4. 협력 시스템 서버

4.1 서버 모델

제안된 협력 시스템은 산업 디자인 협력 작업을 지원하며, 확장이 용이하도록 사용자 관리 기능과 세션 관리 기능을 분리하였다 [그림 5]. UMS는 사용자의 접속과 타 사용자 접속 및 메시지 전달 등의 협력 작업 이전의 서비스를 지원하며, 세션 생성 요청을 받아 세션 관리 서버에게 서비스를 요청하고 그 결과를 참여자에게 알린다. 그리고, SMS는 현재 디자인 프로젝트 스케줄을 관리하며, 협력 작업이 수행되는 세션의 관리와 협력 작업 시 발생되는 공유 데이타의 관리를 위한 이벤트 처리 서비스를 제공한다. 그리고, 서버의 확장성을 지원하기 위해 중앙의 메인 서버인 IS를 제공하여, 사용자 서버간 이벤트 교환 및 공유 데이타의 교환을 가능하게 하여 협력 환경에 따라 확장이 용이하도록 하였다.

서버 가 모듈의 기능은 다음과 같다. UMS는 사용자를 관리하며, 사용자간의 메시지 전송을 담당한다. 또한, 사용자의 세션 서비스(생성, 파괴, 열람, 참여) 요청을 받아, SMS에게 전송한 후, 그 결과를 사용자에게 제공한다. 그리고, IS에 접속하여 다른 UMS에 접속된 사용자간의 정보 교환 서비스를 제공한다. SMS는 세션에 대한 서비스를 UMS에게 제공하며, 협력 작업에 관한 데이타를 저장하고 있다. UMS로부터 세션 서비스 요청

을 받으면, 서비스 가능 여부를 판단하여 UMS에게 그 결과를 돌려준다. 이때, 서비스 요청을 한 사용자에게 고유의 키 (key)를 제공하여, SMS에 접속 시 사용자 아이디, 암호, 키, 세션 접속 포트의 4가지가 일치하여야만 접속이 가능하도록 하였다. 키의 경우, 난수 발생기에 의해 서비스 요청을 한 사용자에게 UMS가 SMS의 출력을 받아 이를 사용자에게 제공하기 때문에, USM로부터 인증 받지 못한 사용자의 SMS 접근을 차단하였다. SMS에서는 공유 객체의 일관성 유지하기 위해서 낙관적 로킹 알고리즘이 사용된다[4][5]. 그리고 관계형 데이타 베이스와 연동되며, 우선순위 큐(priority queue) 스케줄링 기법을 사용하여 이벤트를 처리한다.

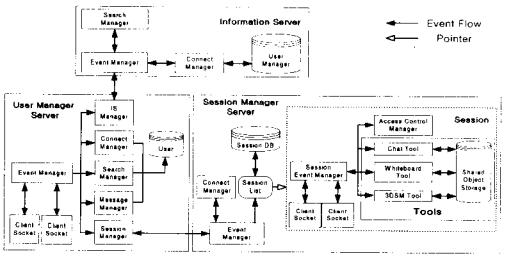

그림 5 협력 시스템 서버 구조

각 서버들은 같은 시스템뿐만 아니라, 다른 시스템에서도 작동이 가능하며 각 서버들은 [그림 6]과 같이 확장 가능하다. 공유 데이타의 조작으로 부하가 집중될 수 있는 SMS를 다수로 두어 작업 부하를 분산시킬 수 있다. 그리고, IS에 의해 다수의 UMS 정보 공유 방법을 제공한다.

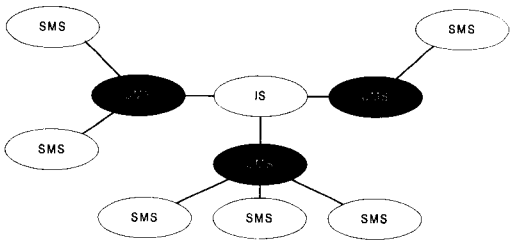

그림 6 확장된 시스템 구조

또한, 서버와 클라이언트간의 이벤트 처리와 통신 방법을 위한 모듈을 개발하였다. 이 모듈은 협력 시스템을 구성하는 가장 하부 모듈로, 이를 바탕으로 다양한 서비스를 제공하는 서버 개발이 가능하다. 그리고, 다수의 플랫폼을 지원하기 위해 서버 플랫폼 독립적인 Java JDK 1.2.1을 사용하였으며, 관련 클래스를 Java의 패키지 형태로 묶어 사용과 유지 보수를 용이하게 하였다.

4.2 공유 데이타의 처리 방법

세션에서 발생되는 이벤트는 SMS에 의해 관리된다. SMS는 [그림 7]과 같이 공동 작업 시 사용되는 공유 객체의 메타 데이타를 저장하며, 필요 시 멀티미디어 데이타를 저장한다.

클라이언트는 서버에 접속되어 세션을 생성, 참여, 관람 등과 같은 서비스를 세션 관리자로부터 제공받고 산업 디자인 프로세서 기반의 디자인 협력 진행 상태에 대한 서비스를 제공받는다. 세션에 참여한 참여자들은 peer-to-peer 방식의 연결 구조를 가지며, 협력 작업중 발생되는 공유 데이타 중 메타 정보는 서버에 의해 관리되며, 중요도에 따라 공유 데이타는 서버에 의해 참여자에게 분배되거나 직접 전송된다.

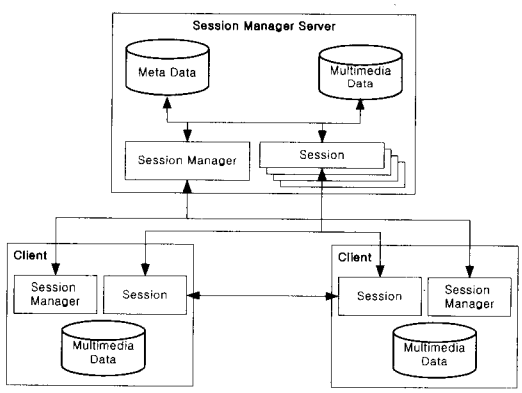

그림 7 공유 객체 관리

4.3 이벤트 처리 모듈

이벤트 처리 모듈은 클라이언트의 이벤트 및 자체에서 발생되는 이벤트를 처리 할 수 있도록 설계되었다. 이 모듈은 협력 시스템에서 빠른 응답성 요구사항을 만족하기 위해 서로 충돌되지 않는 이벤트간의 병행처리를 지원하며, 서비스의 추가를 용이하게 하여 산업 디자인 프로세스뿐만 아니라 타 협력 시스템의 이벤트 처리 방법으로 사용될 수 있는 구조를 갖는다. 이벤트 처리를 위해 이벤트를 정의하는 이벤트클래스와 이벤트 처리를 수행하는 EventThread 클래스와 우선 순위에 따른 이벤트 큐를 가지는 EventMonitor 클래스를 개발하였다. [그림 8]은 이벤트 처리 모델로, 생성된 이벤트는 이벤트 모니터의 우선순위 큐에 이벤트가 삽입되고, 이베트

처리부에 의해 처리되는 구조를 가진다.

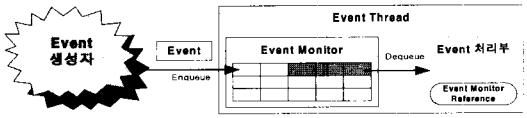

그림 8 이벤트 처리 모델

이벤트 쓰레드는 이벤트 모니터의 큐가 비어 있을 때 쓰레드를 수면(sleep)시키고 큐가 비어 있지 않으면 이를 처리하도록 하여 자원 소모를 최소화한다 [그림 9].

```
1: While (true)
2:if EventQueue is Empty then
3:   Sleep Event Thread
4:else
5:   Dequeue Event
7:   Wake up Event Thread
6:   Event Processing
```

그림 9 이벤트 쓰레드 알고리즘

[그림 10]은 이벤트클래스 다이어 그램으로 이벤트를 정의하는 CEvent 클래스와 큐의 역할을 수행하는 EventMonitor 클래스와 이벤트를 처리하는 Event Thread 클래스로 구성되어 있다.

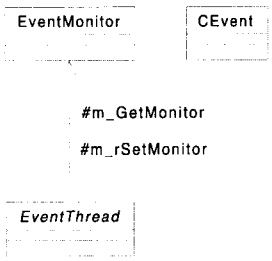

그림 10 이벤트클래스 다이어그램

다수의 이벤트 쓰레드가 동시에 다양한 작업들을 처리하기 위해 이벤트 생성자는 이벤트 쓰레드가 될 수 있다. 즉, 이벤트 모니터 참조에 의해 이벤트 쓰레드가 다른 이벤트 쓰레드에서 이벤트를 전파시킬 수 있다.

이러한 구조는 [그림 11]과 같이 한 개의 이벤트 쓰레드가 다수의 이벤트 모니터의 참조를 가지고 이벤트에 따라 서비스 해주는 이벤트 쓰레드의 이벤트 모니터에게 이벤트디스패치 기능을 제공하여 서로 충돌이 발생되지 않는 이벤트의 병행 처리 방법을 제공한다.

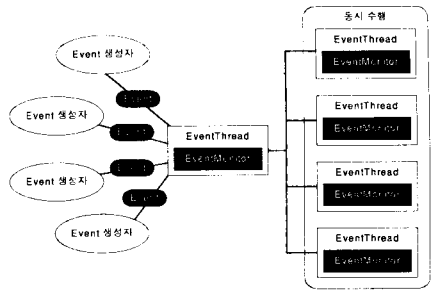

그림 11 확장된 이벤트 처리 모델

4.4 사용자 관리서버 (User Manager Server)

UMS는 사용자의 접속을 관리하고 SMS에 접속되어 사용자에게 세션 서비스를 제공하며, IS에 접속되어 다른 UMS와 정보를 교환한다. UMS는 클라이언트 관리 모듈과 사용자간 메시지 교환과 사용자 정보 검색 모듈과 SMS와 IS와 통신 모듈로 구성되어 있다. 각 모듈들은 이벤트 처리 모듈과 통신 모듈을 사용하여 [그림 12]와 같은 형태로 되어 있는데 다수의 이벤트 쓰레드에 의해 병렬적으로 이벤트가 처리되는 구조로 되어 있다. 사용자 이벤트관리자는 통신 모듈로부터 모든 이벤트를 입력받아 처리 이벤트 쓰레드에게 이벤트를 디스패치 한다. 이 모듈은 다른 이벤트 쓰레드에게 이벤트를 전달하여야 하기 때문에 다른 이벤트 쓰레드보다 높은 우선 순위를 갖는다. 처음 접속한 사용자는 연결 관리자에 의해 사용자 인증 과정을 수행하게 된다. 인증을 받은 사용자는 메시지 관리자에 의해 실시간 메시지를 전송 서비스와, 검색 관리자의 사용자 정보를 검색 및 세션 관리자에 의해 세션의 서비스를 요청할 수 있다.

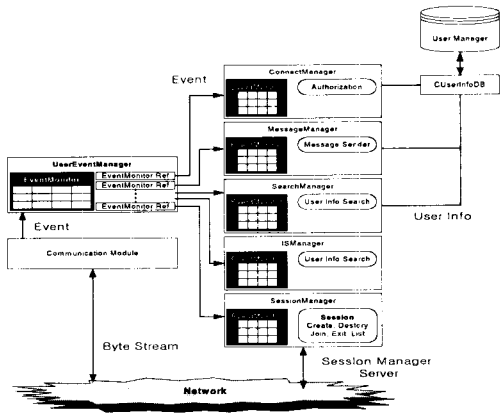

그림 12 UMS 구조

[그림 13]은 UMS의 클래스 다이어그램으로 이벤트 쓰레드를 상속받아 이벤트를 디스패치 해주는 User EventManager, 사용자 인증을 위해 데이타 베이스와 접속하여 사용자 정보를 검색하는 ConnectManager, 사용자간 메시지 교환을 위한 MessageManager, 사용자 검색을 위한 SearchManager, SMS에 접속되어 세션 서비스를 제공하는 SessionManager, IS에 접속되어 서비스하는 ISManager가 있으며, 접속된 사용자 목록을 저장하는 ConnectClientList로 구성되어 있다.

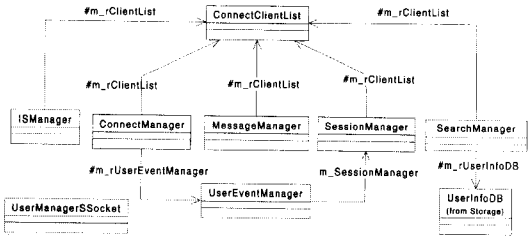

그림 13 UMS 클래스 다이어그램

4.5 세션관리자 서버 (Session Manager Server)

SMS는 크게 디자인 프로젝트의 스케줄 정보를 저장하고 이 참여자에게 제공하는 기능과 스케줄에 따른 협력 작업 가상 공간인 세션을 지원하는 기능을 제공한다. 주요 모듈은 이벤트 처리 모듈과 통신 모듈을 사용하여 [그림 14]와 같이 개발되었는데, 다수의 이벤트 쓰레드에 의해 병렬적으로 이벤트가 처리되는 구조를 가진다. 관리자 모듈들은 이벤트 쓰레드를 상속하여 구현된다. 세션 관리자는 세션들을 생성, 파괴, 참여의 서비스를 UMS에게 제공한다. 생성된 세션은 새로운 포트가 할당되어 사용자의 접속을 기다린다.

세션에서 사용되는 협력 응용 프로그램들 또한, 이벤트 쓰레드를 상속받아 구현되며 자신이 처리해야할 이벤트에 대한 정보를 가지고 있어 이벤트관리자가 협력 응용 프로그램들의 리스트를 순환하면서 처리 모듈을 찾는 방법을 사용한다. 이러한 방법은 차후 협력 응용 프로그램의 추가 시 이벤트정보와 처리 모듈을 가지는 클래스를 리스트에 삽입하는 단순한 과정만이 필요하기 때문에 매우 쉽게 협력 응용 프로그램을 추가할 수 있다.

[그림 15]는 SMS의 클래스 다이어그램으로 크게 이벤트 쓰레드를 상속받은 클래스와 Tool Thread를 상속받은 클래스로 나누어진다. 이벤트 쓰레드는 세션 서비스를 위한 기본적인 기능을 수행하고, Tool Thread는

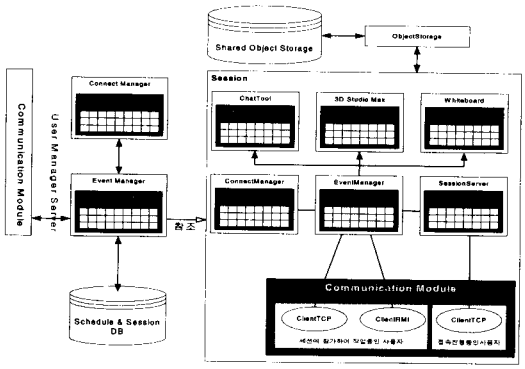

그림 14 SMS 구조

협력 작업에 사용되는 응용 프로그램의 공유 데이타 처리를 수행한다.

SessionManagerEventManager는 UMS로부터 요청받은 이벤트를 처리한다. SessionManagerConnectManager는 UMS의 접속을 관리하며 SessionConnectManager는 생성된 세션에서 참여하는 참여자의 접속을 관리하고 SessionEventManager는 세션에서 발생되는 이벤트를 처리한다. ConnectClientIPList는 세션에 접속된 참여자의 목록을 관리한다. SessionInfo는 생성된 세션의 제목, 목적, 사용 도구 목록 등의 정보가 저장되며, 세션은 사용자의 요구에 따라 세션 초기화 작업을 수행하고, 사용자의 접속을 기다린다. 3D StudioMax, ChatTool, 화이트보드는 클라이언트가 발생시키는 이벤트를 필요에 따라 데이타 베이스에 기록하고, 세션에 참여한 사용자에게 이벤트를 전달한다.

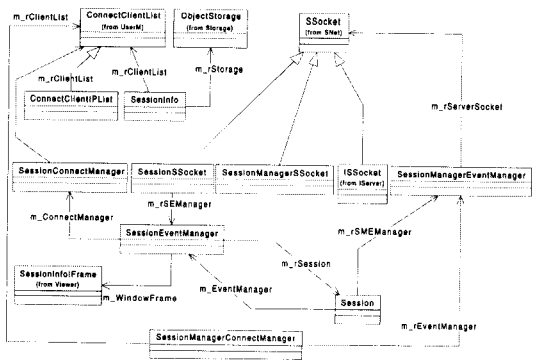

그림 15 SMS의 클래스 다이어그램

4.6 정보서버 (Information Server)

IS는 UMS간의 연동 방법을 제공하며 [그림 16]과

같은 구조를 가진다. 이벤트관리자는 UMS로부터 서비스 요청을 받아 이를 처리한다. IS는 접속된 UMS의 정보를 저장하고 있으며 이를 바탕으로 정보 교환 방법을 제공해준다.

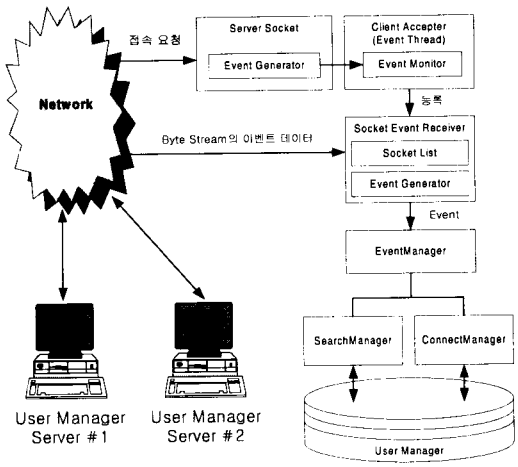

그림 16 Information Server 구조

[그림 17]은 IS의 클래스 다이어그램을 나타낸 것이다. UMS를 관리하는 ISConnectManager와 이벤트를 처리하는 ISEventManager와 UMS간의 서비스를 제공하기 위한 ISSearchManager가 있으며, 접속된 UMS의 정보를 저장하기 위한 ConnectUserMList가 있다.

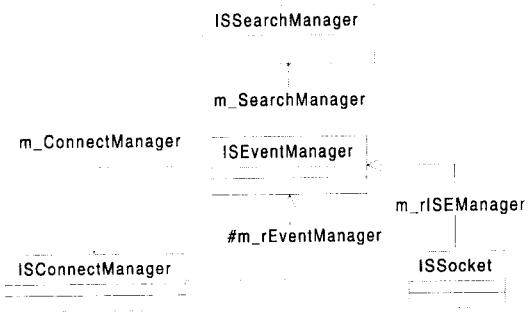

그림 17 IS의 클래스 다이어그램

4.7 패킷 정의

서버간과 서버와 클라이언트가 통신을 위해 [그림 18]과 같이 패킷의 CommandID와 OptionID를 정의하여 서비스를 구분하는 2가지의 명령어를 가진다.

그리고, CommandID와 OptionID를 사용하여 [표 1]과 같이 서버와 서버 사이의 통신과 서버와 클라이언트 간의 통신 프로토콜을 정의하였다. 또한, 협력 응용 프로그램에서 발생되는 공유 데이타의 조작 명령을 전송하기 위해 CommandID와 OptionID를 정의하여 사용하는 구조를 가지도록 하였다. CommandID와 OptionID는 4Byte로 이후 추가되는 어떠한 협력 응용도 지원할 수 있을 만큼 충분한 크기를 가진다. 정의한 패킷은 생성자와 처리될 위치 및 명령(데이타)을 모두 가지고 있기 때문에 처리가 용이하며 이후 추가되는 협력 응용 프로그램의 객체 공유를 위한 방법을 제공해준다.

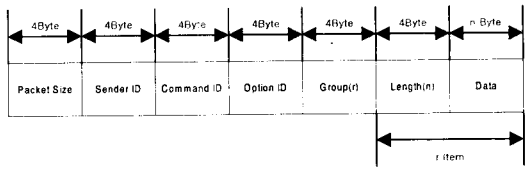

그림 18 패킷 구조

표 1 패킷 설명

	설명
Packet Size	패킷 전체 크기
Sender ID	보낸 사람의 IP Address
Command ID	패킷이 처리되어야 할 모듈에 대한 정보
Option ID	패킷 처리시 사용되는 옵션
Group	Item의 개수를 나타낸다.
Length	Data의 길이 정보
Data	실제 데이타

4.8 통신 모듈

통신 모듈은 서버간 통신을 지원하며 클라이언트의 접속을 유지 관리한다. 통신은 소켓을 통한 TCP 방법과 Java의 RMI(Remote Method Invocation) 방법을 사용한다. TCP 방식의 경우 Java이외의 개발 언어와 호환이 용이하며 RMI 방법은 Java 언어를 사용하여 분산 객체 제어 방법을 지원한다.

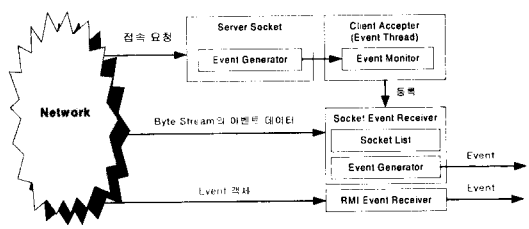

그림 19 클라이언트 접속 및 이벤트 처리

통신 모듈은 이벤트 처리 모듈을 사용하여 클라이언트의 접속 및 클라이언트에서 발생되는 이벤트를 처리 모듈에게 제공한다. 소켓 접속을 위해서 서버 소켓은 클라이언트의 접속 요청을 받아 클라이언트 억셉터(client accepter)에게 제공한다. 클라이언트 억셉터는 클라이언트의 관리를 위해 초기 설정을 하며 소켓 이벤트 수신자에게 이벤트 수신을 위해 클라이언트를 등록한다. 소켓 이벤트 수신자는 클라이언트로부터 수신된 바이트 스트림(byte stream)의 이벤트 데이타를 이벤트객체로 변환한 후 이벤트처리 모듈에게 제공한다. RMI 접속을 위해서 RMI 이벤트 수신자에 의해 이벤트객체의 송·수신이 처리된다 [그림 19].

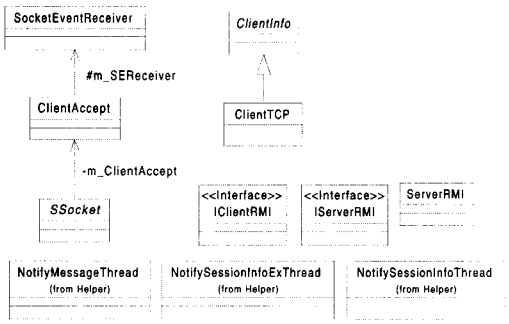

그림 20 통신 모듈 클래스 다이어그램

[그림 20]은 통신 모듈의 클래스 다이어그램으로 ClientAccept은 EventThread를 상속받아 구현되었고 SSocket은 이벤트를 생성하는 생성자가 되며 ClientAccept는 이벤트를 처리하는 소비자가 된다. ClientAccept는 사용자의 접속 시 생성되는 소켓의 초기화 과정을 수행하며 ConnectManager의 Event Monitor에게 클라이언트의 접속 사실을 이벤트클래스를 생성하여 알린다. SSocket은 ClientAccept 클래스를 가지며 클라이언트의 접속 시 ClientAccept의 Queue에 클라이언트 소켓을 삽입하여 소켓이 초기화 작업을 수행한다. ClientInfo는 접속 사용자의 소켓 레퍼런스와 사용자를 증명하는 ID, 접근 권한 레벨 등의 정보를 가지며 서비스 요청 시 ClientInfo를 모든 이벤트클래스는 이를 참조하여 서비스 요청자를 명확하게 명시해준다. ClientTCP와 ClientRMI는 ClientInfo를 상위 클래스로 실제 데이타 입출력 함수가 구현되어 있다. SocketEventReceiver는 소켓으로 입력받은 바이트 스트림을 이벤트객체로 만들어 EventThread의 Event Monitor에 삽입한다. UserManagerSSocket은 UMS의 서버 소켓이며, SessionManagerSSocket은 SMS의 서버 소켓이고, SessionSSocket은 세션의 서버 소켓이며, ISSocket은 IS의 서버 소켓이다. NotifyMessage Thread는 특정 사용자에게 쪽지를 보낼 때 사용되고 NotifySessionInfo ExThread는 세션 정보를 전송하는 쓰레드 클래스이다. NotifySessionInfoThread는 세션중 도구가 한정된 대화방에 대한 목록을 전송하는 클래스이다.

4.9 공유 객체 영속성을 위한 데이타 베이스 연동

공유 객체의 영속성과 디자인 프로세스에서 요구되는 스케줄 정보를 저장하기 위해 JDBC를 이용하여 데이타베이스 연동 모듈을 [그림 21]과 같이 개발하였다.

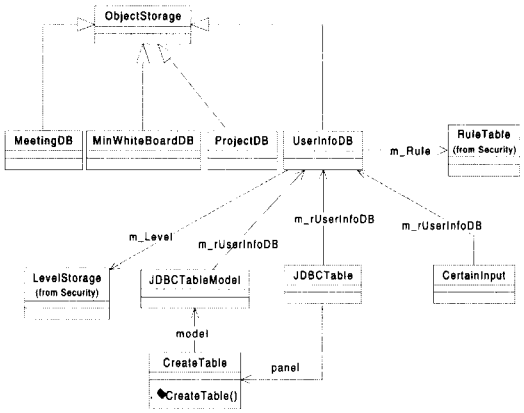

그림 21 Storage 클래스 다이어그램

MS Access와 Java간의 한글 호환 문제를 해결하고, DB 접속과 종료 함수와 같은 일반적인 함수가 구현된 CObjectStorage 클래스를 생성하고 이 함수를 상속받아 사용자 정보를 관리하는 CUserInfoDB 클래스와 프로젝트 정보를 관리하는 CProjectDB 클래스, 회의 정보를 관리하는 CMeetingDB 클래스, 세션에서 사용된 화이트보드의 이벤트를 저장하기 위한 C화이트보드DB 클래스, 3D Studio Max의 이벤트를 저장하기 위한 C3DSMaxDB가 있다. 그리고, 사용자 등록 시 입력 내용을 검사하기 위한 CertainInput 클래스가 있다.

4.10 사용자 인터페이스

서버에 상태를 모니터링 하기 위해 Java의 스윙을 사용하여 [그림 22]와 같이 사용자 인터페이스를 개발하였다.

이벤트를 모니터링하기 위해 이벤트 모니터의 큐

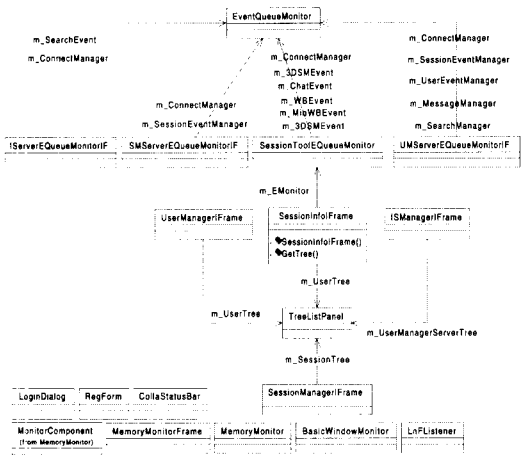

그림 22 View 클래스 다이어그램

(queue)의 상태를 출력하는 EventQueueMonitor를 개발하였으며, 이를 사용하여 각 서버별 이벤트 쓰레드의 상태를 출력하는 UMServerEQueueMonitorIF, SMS ServerEQueueMonitorIF, ISServerE- QueueMonitor IF가 있다. 그리고, 서버에 접속한 사용자 및 서버 목록을 출력하기 위한 TreeListPanel이라는 트리 컨트롤이 있으며 이를 사용하여 UMS에 접속된 사용자 정보를 출력하는 UserManagerIFrame, SMS에서 접속된 UMS의 목록 정보를 출력하는 SessionManager IFrame, IS에서 접속된 UMS의 목록을 출력하는 USManagerIFrame를 개발하였다. 그리고, 각 서버별 메모리 사용량을 출력하는 MemeryMonitor와 Memory Monitor Frame가 있다.

5. 협력시스템 클라이언트

클라이언트 시스템은 서버로부터 디자인 프로세스에 의한 스케줄 정보를 획득하여 현재 프로젝트의 진행상황을 인지하고 다수의 참여자와 디자인 협력 작업을 지원한다. 이를 위해 클라이언트 시스템은 [그림 23]에서 음영부분인 협력 작업 준비 단계 모듈과 협력 작업 단계로 나누어져 있다.

협력 작업 준비 단계에서는 USM에 접속하여 타 사용자들의 접속유무와 협력 작업 진행 상황들을 사용자에게 제공한다. 이는 통신 관리자를 통해 입력된 패킷의 이벤트 객체로 만들어 세션 관리자와 사용자 관리자에서 처리하여 사용자 인터페이스를 통해 제공된다. UMS는 타 사용자의 접속 유무에 대해 정보를 제공하며 사용자 검색, 사용자에게 메시지 보내기 등의 서비스를 제공한다. 세션 관리자는 현재 협력 작업 상황을 제공하며 세션 생성, 참여 서비스 요청 방법을 제공한다. 가상공간 뷰어 (Virtual Space Viewer)는 사용자 정보 및 세션에 대한 정보를 출력하는 사용자 인터페이스 모듈이다.

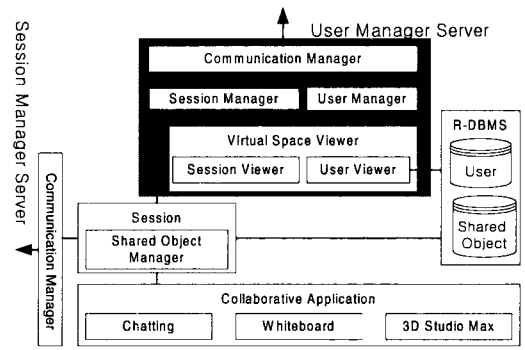

그림 23 협력 시스템 클라이언트

협력 작업 단계는 세션 생성 또는 참가를 통해 SMS에 접속하여 타 사용자와 데이타를 공유하여 작업을 수행함을 의미한다. 세션의 참여는 UMS로부터 접속 인증을 받은 후 세션의 접속에 필요한 정보를 수신하여 SMS에 접속한다. 세션 모듈은 세션의 참여시 필요한 정보를 SMS로부터 전송 받아 세션에 접속 기능을 수행하고 협력응용 모듈은 협력 작업 시 사용되는 응용 프로그램에 대한 인터페이스를 제공한다. 현재 구현된 응용 프로그램은 채팅, 화이트보드, 3D Studio Max가 있다.

5.1 이벤트 처리 및 통신 모듈

클라이언트의 이벤트 처리와 통신 모듈은 [그림 24]와 같다. 서버와는 달리 이벤트를 정의하는 클래스와 이벤트를 처리하는 프로토콜 클래스로 나누어진다. 프로토콜 클래스는 이벤트를 처리할 수 있는 구체적인 코드를 담고 있다.

CEvent는 서버와 통신하기 위해 정의된 패킷을 이벤화한 클래스로 객체와 데이타 정보를 갖고 있다. CCollaSocket은 CReadPacket과 CWritePacket을 사용하여 서버와 통신을 하며 CProtocol 리스트를 저장하고 있어 서버로 수신 받은 이벤트를 CProtocol 리스트를 순회하여 처리 모듈을 찾도록 하였다. 이 방법은 새로 개발된 협력 응용을 추가하기 위해 CProtocol을 상속받은 클래스와 사용자에게 객체 정보를 출력하는 인터페

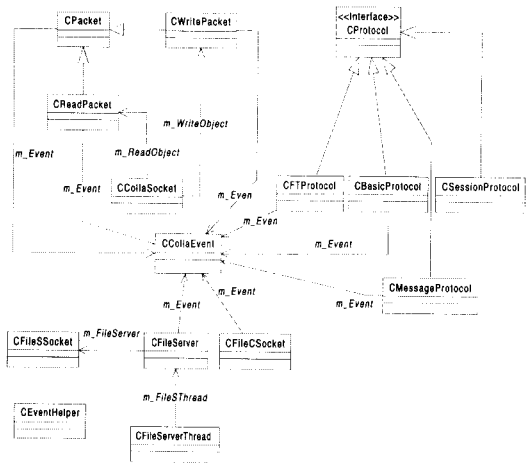

그림 24 이벤트 및 통신 클래스 다이어그램

이스 클래스의 개발만으로 협력 응용을 개발할 수 있는 수단을 제공한다. CProtocol을 상속받아 생성된 CBasicProtocol은 접속에 관련된 이벤트를 처리하고, CSessionProtocol은 세션 이벤트를 처리하며, CMessageProtocol은 사용자간 메시지와 사용자 정보 검색 이벤트를 처리한다. CFTPProtocol과 CFileClient, CFileCSocket, CFileClientThread는 파일 전송시 클라이언트 모듈이며 CFileServer, CFileSSocket, CFileServerThread는 파일 전송시 서버 모듈에 해당한다. CEventHelper는 이벤트를 바이트 스트림으로 전환시 도움을 주는 함수들의 집합이다.

5.2 협력 응용 프로그램 관련 모듈

개발한 협력응용 프로그램들로는 서버에 접속하여, 디자인 프로세스에 따른 스케줄 정보를 수신하여 사용자에게 제공하고 세션 서비스를 제공하는 CSession Manager와 CUserStateView, CToolIconWnd 클래스가 있다. 그리고, 디자인 협력 작업을 지원하기 위해 채팅, 화이트 보드, 3D Studio Max의 객체 공유 모듈도 있다.

CSessionManager는 CSessionProtocol에서 처리되는 이벤트를 사용자에게 제공하고, CUserStateView는 세션에 참여한 사용자의 상태를 제공한다. 그리고 CToolIconWnd는 사용자에게 세션에서 사용되고 있는 도구에 대한 정보를 출력한다.

협력 작업에 사용되는 도구들은 이벤트를 처리하기 위해 CProtocol을 상속하여 개발된 모듈과 사용자 인터페이스 모듈로 개발된다. 채팅 기능을 제공하기 위해 채팅 사용자 인터페이스인 ChattingFrame과 채팅 이벤트를 처리하는 ChattingProtocol을 개발하였고, 2D 이미지 객체 공유를 지원하는 화이트 보드는 이벤트 처리를 위한 CWBProtocol과 2D 이미지 객체를 정의한 CPic와 2D 도형 정보를 정의한 CLinePoint, CErase 클래스를 개발하였다. 사용자 인터페이스 클래스인 C화이트보드Tool을 개발하였고, 3D Studio Max의 plug-in과 공유 메모리로 객체 정보를 송·수신 기능을 제공하기 위해 C3DSMFrame 클래스를 개발하였으며, 이벤트 처리를 위한 C3DSMProtocol를 개발하였다.

5.3 3D Studio Max 관련 모듈

산업 디자인 프로세서의 지원을 위해서는 화이트보드를 통한 2D 객체 공유뿐만 아니라 3D 객체의 공유가 요구된다. 제안된 시스템에서는 개방형 구조를 기반으로 객체지향형 프로그래밍 환경을 제공하여 자신이 제공하는 기능을 변경 또는 추가하기 위해 plug-in SDK (Software Development Kit)을 지원하는 3D Studio Max의 공유 plug-in을 개발하였다.

3D Studio Max의 공유 plug-in은 단일 컴퓨터 시스템에서 수행되는 3D Studio Max 작업을 지리적으로 떨어진 분산 환경으로 확장시켜 다수의 사용자가 공동으로 3D 모델링 작업을 수행할 수 있는 환경 제공하여 산업 디자인 협력 작업 시 요구되는 3D 객체의 공유 방법을 제공하도록 하였다.

3D Studio Max의 객체 공유를 위해 3D Studio Max의 객체 정보 입출력을 위해 plug-in을 통해 3D Studio Max에서 생성된 객체 정보에 접근하고, 이를 서버에 송·수신 받기 위해 공유 메모리를 사용한다.

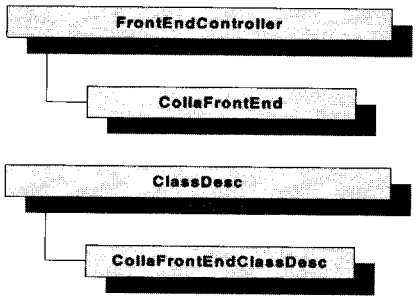

그림 25 Plug-in Main Class

3D Studio Max의 공유 객체 정보를 접근하기 위해 [그림 25]의 클래스들을 사용한다. FrontEndController는 새로운 'Front end' plug-in을 생성할 수 있는 Base Class이다. Max의 전반적인 사용자 인터페이스를 컨트롤할 수 있으며 Max의 기본적인 폼을 사용한 응용

애플리케이션의 생성을 가능하게 한다. 사용자 인터페이스에는 Title Bar, Drop-Down Menus, Toolbar, Command Panel, Time Controls, Right-Click Viewport Label Menu, Right-Click Object Menu등이 있다. CollaFrontEnd는 협력 작업 환경에 적합한 Max의 사용자 인터페이스를 구성하기 위한 base class이다. Max 구동 시 Plug-in 초기화 작업을 수행하며 상위 클래스인 FrontEndController의 함수들을 이용한 사용자 인터페이스 이벤트를 인터럽트 한다. 주로 사용자의 모델링 데이타를 전송하기 위하여 이벤트 발생기 역할의 Right-Click Object Menu를 재 설정한다. 메뉴의 초기화 작업 및 메뉴 선택 이벤트 발생에 대한 처리 루틴을 포함하며 Viewport상의 오브젝트 제어권을 갖는다. ClassDesc는 Max의 plug-in은 필수적으로 이 클래스를 포함한다. Class Descriptors은 DLL에서 Plug-in 클래스에 대한 정보를 시스템에 제공한다. 이 클래스 메소드를 통하여 종류별, 기능별 Plug-in Class Descriptor를 생성한다. CollaFrontEndClassDesc는 개발 Plug-in 정보를 상세히 기술하는 class로써 클래스명과 다수의 Plug-in과 구별할 수 있는 유일한 클래스 ID(SuperClassID & ClassID), Command Panel의 카테고리이름 등의 함수들을 제공한다.

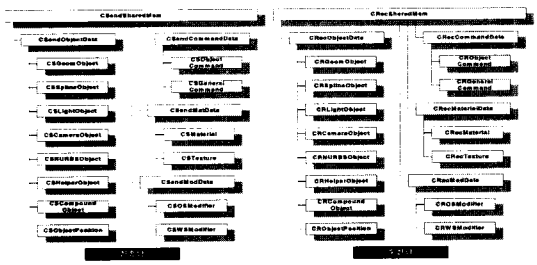

그림 26 공유 객체 송·수신을 위한 공유 메모리 관리 클래스

Plug-in을 통해 3D Studio Max의 공유 객체를 공유 메모리를 통해 통신 모듈과 송·수신하기 위해 [그림 26]과 같이 정의하였다. 3D 객체의 전송 부분과 수신 부분의 둘로 나눈다. 전송단과 수신단의 구조는 매우 유사하며 주로 전송단에서는 공유 데이타의 전송 순서를 정립하여 공유 메모리에 기록하며 수신단에서는 전송된 데이타를 공유 메모리를 통하여 순서대로 읽어 오는 역할을 담당한다.

CSendSharedMem/CRecSharedMem는 공유 모델링 객체 데이타 관리 부분의 최상위 클래스이며 하위 실질적 데이타 분석 클래스의 공통 처리 모듈을 포함한다. 또한 객체의 ParameterBlock에 대한 엑세스 제어권을 가지며 공유 메모리의 실질적 관리를 담당한다. CSendObjectData/CRecObjectData는 Max의 오브젝트(Primitive Object, Extended Primitive Object, Shape Object, Compound Object, NURBS Object, Helper Object, Carema Object, Light Object. etc)에 대한 공통 분석 모듈을 포함한다. CSendModData/CRec ModData는 Max의 Modifier(Object Space Modifier, World Space Modifier)에 대한 공통 분석 모듈을 포함한다. CSendMatData/CRecMaterialData는 Max의 Material과 Texture 대한 공통 분석 모듈을 포함한다. CSendCommandData/ CRecCommandData는 Max의 내부/외부 명령어들(File Loading, Viewport 제어, 초기화, 랜더링 등)에 대한 공통 분석 모듈을 포함한다.

6. 시스템 구현 및 성능평가

구현 시스템은 채팅, 화이트보드와 3D Studio Max를 협력 도구로 사용하는 디자인 협력 작업을 지원한다. 서버 시스템은 서버에 접속한 다른 서버나 클라이언트 목록을 표시해주며, EventMonitor의 큐의 상태를 출력하여 서버의 상태를 모니터링 한다. 만약 큐의 이벤트가 일정 시간 이상 처리되지 않으면 처리 스레드 모듈이 오류 발생을 알 수 있으며 처리되어야 할 이벤트가 많다면 해당 모듈이 서비스해주는 모듈의 개선 작업이 필요함을 알 수 있다.

6.1 서버 구현

서버는 IS, UMS, SMS로 구성되며 각각은 스윙(swing)을 사용하여 서버의 상태를 관리자에게 제공할 수 있는 사용자 인터페이스를 제공하였다. [그림 27]은 IS의 구현 화면으로 IS에 접속된 UMS의 목록과 현재 메모리 사용 상태와 이벤트 모니터에서 처리를 기다리고 있는 이벤트의 수를 출력하여 서버의 상태를 모니

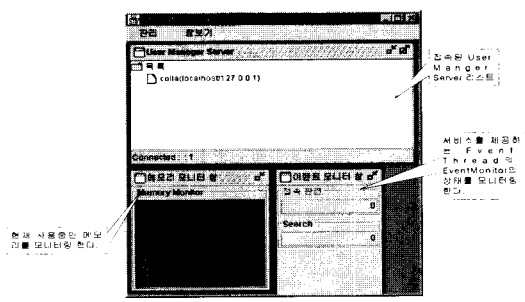

그림 27 Information Server 구현 화면

터링 한다.

[그림 28]은 UMS의 구현 화면으로 접속된 사용자의 목록, 메모리 사용 상태, 사용자 관리를 위한 메뉴, SMS와 IS에 접속을 위한 사용자 인터페이스를 제공한다.

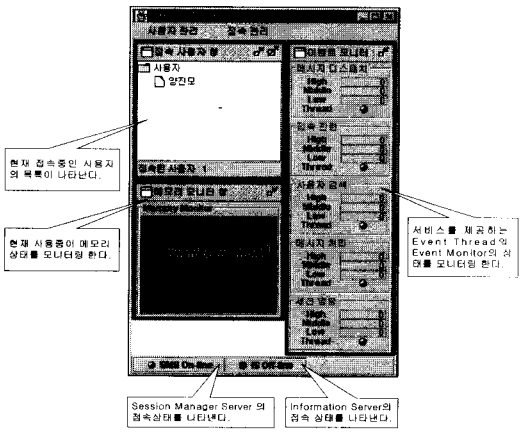

그림 28 UMS 구현 화면

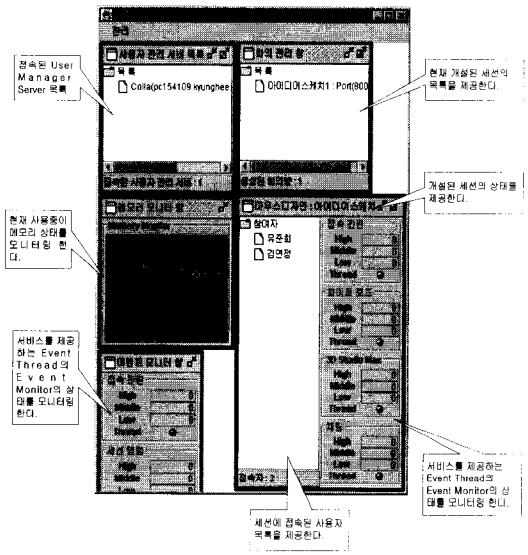

그림 29 SMS 구현 화면

[그림 29]는 SMS의 구현 화면으로 접속된 UMS의 목록, UMS로부터 세션 서비스 요청을 위해 입력된 이벤트의 처리 상태를 출력하는 이벤트 모니터창, 현재 진행중인 세션의 목록을 출력하는 창, 세션에 접속된 사용자의 목록과 세션 참가자들이 발생시킨 이벤트를 모니터링하는 창과 현재 메모리 사용 상태를 출력하는 창으로 구성되어 진다.

6.2 클라이언트 구현

클라이언트 소프트웨어는 산업 디자인 프로세스를 사용자에게 제공하여 협력 작업의 진행 상황을 작업자에게 제공하고 공동의 협력 작업을 위한 사용자 인터페이스를 제공한다. [그림 30]의 좌측 상단에 출력된 산업 디자인 프로세스 중 현재 진행중인 마우스 디자인 협력 작업에 참여하여 3D Stduio Max를 사용하여 3D 객체를 공유한다. 참여자들은 3D 객체의 속성(이동, 재질·색·크기 변경) 변경과 객체의 생성, 삭제 등을 통해 의사를 교환한다. 화이트 보드는 랜더링된 기존의 마우스 이미지의 공유 방법을 제공한다. 또한, 채팅을 통해 객체 변경으로 전달할 수 없는 내용을 참여자들에게 제공한다.

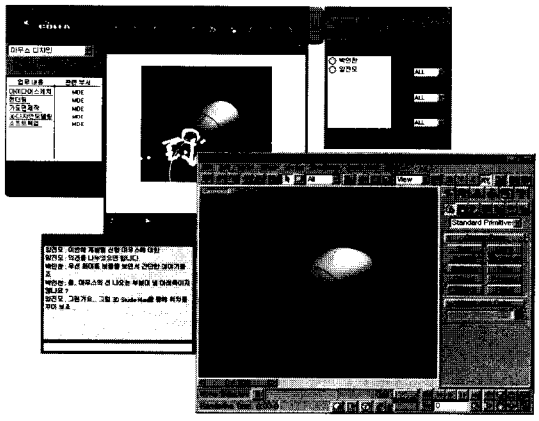

그림 30 클라이언트 구현 화면

6.3 성능 평가

서버의 성능을 평가하기 위해 사용된 시스템은 Pentium II 300Mhz와 224MB RAM 시스템에서 Windows NT 4.0 OS이며, 100Mbps LAN 환경에서 각각 단일 UMS와 SMS에 접속자 수와 이벤트의 발생 빈도에 따른 이벤트의 응답 시간을 검사하였다. 이벤트의 응답 시간은 협력 시스템 개발 시 해결해야할 중요한 문제이다. 그렇기 때문에 서버의 용량 및 각 서버의 응답 시간을 평가하여 각 작업에 적합한 협력 시스템의 구성을 제시하고 한다.

본 논문에서 검사한 성능 평가의 결과를 바탕으로 협력 작업 사용자와 발생되는 이벤트를 바탕으로 이벤트의 발생 빈도보다 사용자의 접속이 많을 경우 다수의 UMS가 요구되며, 접속 사용자에 비해 발생되는 이벤트

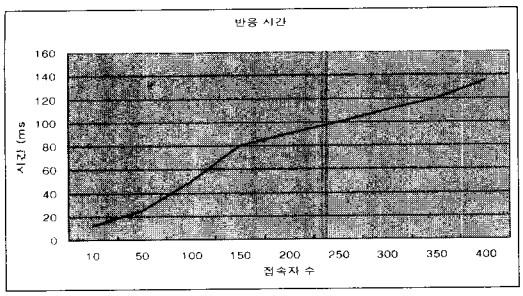

그림 31 UMS의 접속자에 따른 반응 시간

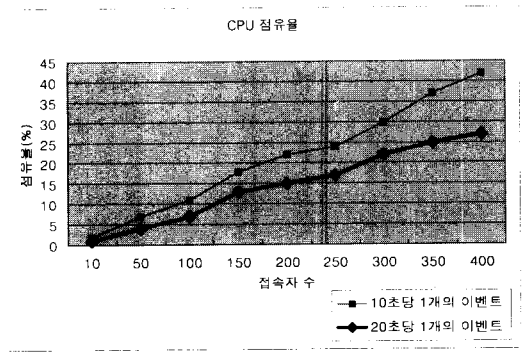

그림 32 이벤트 빈도에 따른 CPU 점유율

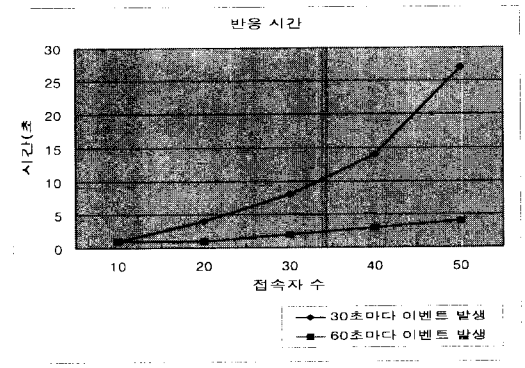

그림 33 SMS의 접속자수에 따른 반응 시간

가 많을 경우 다수의 SMS가 요구됨을 확인할 수 있다.

[그림 31]은 접속자의 수에 따른 UMS의 반응 시간을 나타낸 것으로, 접속자는 10초당 1개의 이벤트를 발생시킨다. 이때, 접속자수에 따른 반응 시간은 선형으로 증가됨을 알 수 있다. [그림 32]는 접속자 수에 의한 CPU 점유율로 이벤트의 발생 빈도의 증가에 따라 비례로 증가함을 보여준다. [그림 33]은 SMS의 사용자 수에 따른 반응 시간으로 10명이 한 세션에 참여하고 30초와 60초마다 각 사용자가 이벤트를 발생시킨다. 이때, 반응 시간은 참여자의 수보다는 발생되는 이벤트의 빈도에 따라 급격하게 증가하는 것을 알 수 있다. 이는 발생된 메시지를 세션에 참여한 참여자에게 전송하고 데이타 베이스에 저장하는 시간이 증가함에 따라 급격히 반응시간이 증가함을 보여준다. 그리고, 이벤트의 발생 빈도를 30명일 때 8초 이하, 40명일 때 14초 이하, 50일 때 20초 이하로 낮추었을 때 처리되는 이벤트보다 이벤트 모니터에 입력되는 이벤트의 수가 증가하여 서버의 수행 오류를 발생시킨다.

앞에서 살펴본 3가지의 성능 평가는 서버에 접속되는 사용자의 수에 협력 시스템의 서버의 성능 저하보다는 각 사용자가 발생시키는 이벤트 빈도에 따라 서버 성능이 급격하게 저하됨을 알 수 있다. 또한, 공유 데이타를 처리하는 SMS와 처리하지 않는 UMS의 경우 접속자 수와 이벤트 발생 빈도를 비교하였을 때 UMS의 반응 시간이 매우 빠른 것을 확인할 수 있다. 이것은 공유 데이타의 처리 작업시 일관성 유지와 협력 작업 참여자에게 공유 데이타의 변경 이벤트를 네트워크를 통하여 제공하여야 하는 작업에서 많은 처리시간이 소요됨을 확인 할 수 있다.

7. 결 론

본 논문에서는 생명 주기가 짧은 산업디자인 애플리케이션 개발 시 다자간 공동작업을 지원할 수 있는 협력시스템 서버와 클라이언트의 설계와 구현에 대한 경험을 기술하였다. 제안된 시스템의 서버는 협력 작업 스케줄과 공유 데이타를 저장하는 SMS와 사용자 인증과 사용자 정보를 제공하는 UMS와 서버의 확장을 지원하는 IS로 구분하였다. SMS는 개발된 산업 디자인 프로세스를 바탕으로 참여한 사용자에게 협력 작업 스케줄을 제공하여야 하며 협력 작업 내용을 데이타 베이스화하여 저장한다. 또한, 클라이언트와 서버간의 이벤트 처리 방법과 통신 모듈을 정의하고, 이를 바탕으로 혼합형 구조의 확장성을 가진 서버와 클라이언트를 개발하였다. UMS의 경우, 이벤트 발생 빈도의 증가에 따른 반응 시간의 증가는 만족할 만한 수준이며 SMS의 경우, 이벤트 발생 빈도의 증가에 따라 급격한 성능 저하를 확인할 수 있었다. 이는 협력 작업 환경에 따라 다수의 SMS의 필요성을 인식시켜 주었으며, 기능에 의한 서버의 분류 방법은 확장이 용이한 협력시스템의 구축 가능성을 확인 시켜주었다. 클라이언트는 사용자의 편의를 위하여 Windows환경에서 Visual C++로 구현하였으며 3D 객체공유 방법을 제공할 수 있는 plug-in기술을 사용하여 3D Studio Max를 분산환경으로 확장하였다.

향후 연구로는 SMS의 부하를 모니터링 하여 접속된 다른 SMS에게 부하를 분산시키는 방법과, 협력 응용에 따른 공유 데이타의 다양한 일관성 유지 방법에 대한 연구가 필요하다. 또한, 현실성 있는 협력 시스템을 구축하기 위해서 동시성제어, 멀티캐스팅, 보안, 멀티유저 인터페이스, 고장허용, 최적화 등의 문제를 해결 해나갈 예정이다.

참 고 문 헌

[1] Tom Rodden, A Survey of CSCW Systems, Interacting with computers, 1991, vol. 3 no. 3, pp. 319~352

[2] 김창헌, 양재헌. Collaborative Virtual Environment에서의 동시성 제어와 기술, 정보과학회지 제 16권 제 7호, 1998. 7.

[3] 궁상환, 황승구, Collaborative Computing 기술 및 응용, 정보과학회지 제 16권 제 7호, 1998. 7.

[4] M.T. Ozsu and P. Valduriez. Principles of Distributed Database Systems. Prentice-Hall, pp. 327~329, 1998.

[5] S. Bhola, G. Banavar, and M. Ahamad. Responsiveness and consistency tradeoffs in interactive groupware. In Proceedings of 7th ACM Conference on Computer Supported Cooperative Work, November 1998.

[6] Greenberg, S. and Marwood, D. Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface. In Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 207~217, Chapel Hill, North Carolina, October 22-26, 1994, ACM Press.

[7] S. Bhola, B. Mukherjee, S. Doddapaneni and M. Ahamad, Flexible Batching and Consistency Mechanisms Building Interactive Groupware Applications, In Proceedings of the 18th ICDCS, 1998.

[8] J. Munson and P. Dewan. "A Concurrency control Framework for Collaborative System". In Proceedings of the 6th ACM CSCW

[9] Annie Chabert, Ed Grossman Larry Jackson, Stephen Pietrovicz, NCSA Habanero: Synchronous Collaborative Framework and Environment, 1998

[10] Roseman, M. and Greenberg, S. Building Real Time Groupware with GroupKit, A Groupware Toolkit. ACM Transaction On Computer Human Interaction 3(1), March 1996

[11] Philip M. Johnson. Experiences with EGRET: An exploratory group work environment. Collaborative Computing, January 1994

[12] 이광행, 전재우, 오삼권, CW-MAN:효율적인 멀티미디어 공동저작을 위한 혼합형 구조의 공동저작 관리 시스템, 한국 정보 처리학회 논문지, p.1253-1262, 1999.5

[13] 정의현, 박용진, "WWW상에서의 공동작업 시스템의 설계 및 구현", 정보과학회논문지(C), 제 3권 제 4호, pp.384-396, 1997, 8

[14] J.H. Lee, A. Prakash, T. Jaeger, and G. Wu, Supporting multi-user, multi-applet workspaces in CBE, Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'96), pp. 344-353, 1996

[15] G. Smith and T. Rodden, SOL: A Shared Object Toolkit for Cooperating Interfaces, Technical Report CSEG/7/1995, Lancaster University, 1995.

[16] P. Dewan and R. Choudhary, A Flexible and High-Level Framework for Implementing Multi-User User Interfaces, ACM Transactions on Information Systems, Vol. 10, No. 4, pp. 345-380, Oct. 1992

[17] A. Prakash and H.S. Shim, DistView: Support for Building Efficient Collaborative Applications using Replicated Objects, Proceedings of CSCW '94, ACM Press, New York, 1994, pp. 153-64.

[18] 최종명, 김형진, 최재영, CoDraw: 자율 객체를 이용한 웹에서의 유연성있는 CSCW 시스템 설계 및 구현, 정보과학회 논문지(C), 제 5권, 5호, 1999. 10

[19] 조성빈, 김진석, 진성일, CSCW를 위한 분산 객체 공유시스템, 정보과학회 논문지(C), 제 5권 3호, 1999. 6

양 진 모

1998년 2월 한신대학교 정보통신학과 졸업. 2000년 2월 경희대학교 전자계산공학과 석사 졸업. 2000년 현재 경희대학교 전자계산공학과 박사과정 재학중. 관심분야는 분산 시스템, 협력시스템.

이 승 룡

정보과학회논문지 : 컴퓨팅의 실제 제 6 권 제 2 호 참조

전 태 웅

정보과학회논문지 : 컴퓨팅의 실제 제 6 권 제 2 호 참조