한국 CAD/CAM 학회 논문집
제4권 제3호 1999년 9월 pp. 200-209

학술논문

제품 데이터 표준에 기반한 원격회의 지원 시스템

최 영*, 양상욱**

Conference System for CAD Based on Product Data Standard

Young Choi* and Sangwook Yang**

ABSTRACT

This paper presents a 3D-conferencing system as a prototype implementation of concurrent and distributed engineering environment using CORBA-Java and SDAI-Java. The system consists of some client and server objects that communicate with each other via ORB. The server objects use lightweight version of current working draft of SDAI-Java to access STEP product data models, and the client objects provide user-friendly graphical interface. The proposed system can be used for communicating with 3D CAD data between remote designers, manufactures and customers regardless of the H/W or S/W platforms they use.

Wey words : CORBA, Java, ObjectWeb, 3D conferencing system, SDAI-Java, STEP

1. 서 론

인터넷과 웹 기술의 보급은 데이터의 공유와 글로벌 협력을 위한 새로운 소프트웨어 도구의 출현을 가능하게 하고 있으며[1] 불과 3년 전인 1990년대 중반부터 비젼으로서 제시되던 Web-enabled engineering은 다수의 개발자나 사용자가 단지 웹브라우저와 같은 최소한의 클라이언트로 제품의 개발에 참여하거나 정보를 공유할 수 있다는 이점으로 인하여, 짧은 기간동안 급속한 발전을 거듭해왔다. 이 과정에서 CAD 벤더들은 다양한 방식으로 웹을 기반으로 하는 제품 개발을 진행시키고 있고[2], 이러한 시도는 3D CAD 데이터를 VRML로 변환하여 브라우저에서 가시화 한다든지[3,4], DWF, CGM, sat 또는 STEP 등의 포맷으로 된 데이터를 웹에 게시하여 웹 브라우저의 플러그인 프로그램을 통하여 브라우징할 수 있게 하는 방식[5], 클라이언트 요구에 따라 현재의 데이터베이스로부터 검색한 결과에 대한 HTML 문서를 생성하여 전달하는 방식 등으로 진행되어 왔다.

이러한 기술의 발달은 설계 및 생산 분야의 연구자 및 엔지니어들에게 새로운 패러다임을 제시하고 있다. 특히 객체 지향 소프트웨어 개발 기법과 분산화 기법, 생산 제품 정보의 새로운 국제 표준의 제정 및 채택은 분산 엔지니어링 환경에서의 동시공학을 구현하기 위한 기반이 되고 있다. CIM이나 동시공학과 같은 기술의 개념은 꾸준히 확산되어 이미 성숙 단계에 들어서 있으나, 구현 소프트웨어 도구의 부족으로 인해 만족 할 만큼 구현되어 있지는 못한 상태이다. 그러나 현재 다양한 분야에서 네트워크를 기반으로 하는 분산 환경과 동시 공학에 관한 연구가 진행 되고 있다.

Hardwick 등은 가상기업 환경에서의 정보 인프라의 프로토타입을 제안한 바 있다. 이 연구의 목적은 분산 시스템에서 데이터 교환의 표준 규약을 만드는데 있으며 그 핵심 개념은 STEP과 CORBA 표준을 인터넷에 적용하는 것이다. 하부 구조에 있어서는 각각 STEP과 CORBA의 구조를 기술하는 근간이 되는EXPRESS와 IDL을 조합 적용함으로써 이종의 분산 공학 환경에서 응용소프트웨어의 통합을 가능하게 한다[6].

Pahng은 그의 논문에서 네트워크 중심의 설계 환경에서 제품 설계 프로세스에 주안점을 둔 연구를 소개하였다[7]. 제안된 프레임워크는 객체 지향 모델링과 평가의 개념을 분산 네트워크 환경으로 확장한 것이다. 이 논문은 서로 상충되는 목적함수를 가진 문제들이 포함된 협동 작업을 분산 환경을 통해 수행하는 예를 보여주며, CORBA가 분산 모듈간의 상호 교신

*종신회원, 중앙대학교 기계공학부
**학생회원, 중앙대학교 기계설계학과

을 위하여 사용되었다.

분산 환경에서 특징형상 개념을 응용하는 연구도 진행되고 있다. 특징형상 개념은 다양한 응용 프로그램에 대해 다양한 관점을 제공하기 때문에 분산환경에 적합한 개념이라는 점에서 응용의 대상이 되고 있다. Martino 등의 논문에서는 특징형상에 기반을 둔 중립적 표현을 이용하여 분산 환경에서 설계와 해석을 통합하는 예를 볼 수 있다[8]. 그의 연구에서는 특징형상을 기반으로 하는 분산 객체지향 시스템 구조에서 특징형상 기반 설계 및 특징형상인식을 동시에 지원한다. Han과 Requicha는 형상 모델러를 감싸는 API를 작성해 분산 환경에서 동작하는 특징형상 인식 시스템을 연구하였고[9], Gadh와 Sonthi는 인터넷을 기반으로 하는 설계에서 특징형상 인식 기법에 의한 다단계의 형상 추상화를 이용하는 연구를 수행하였다[10]. 그러나 인용한 이들 특징 기반의 시스템들은 애플리케이션들 간의 통신 수단으로써 CORBA를 사용하지는 않고 있다.

기업간, 또는 기업내에서 설계 및 생산 정보를 공유하고 관리하는 일은 현재와 같은 인터넷 시대에서 매우 큰 관심사가 되고 있다. Dong과 Agogino는 설계문서와 설계데이터의 관리를 네트워크 환경에서 통합하고 클라이언트-서버 구조를 통해 접근 할 수 있는 프레임워크를 제안하였다[11]. RISESTEP 프로젝트는 이종의 분산환경에서 제품 데이터를 공유하고 공동 협업시스템을 구축하는 것을 목표로 하고 있다. 이 과제에서는 여러 협력자간에 표준화된 제품 데이터를 확장된 기업 환경에서 공유함으로써 공동 협업을 실현하려 하고 있으며, CORBA와 STEP에 기반을 두고 있다[12]. 이와 유사한 시도로 Marache 등은 VR을 이용한 접근 방법에 대해 논의하였다[13].

스탠포드의 SHARE 프로젝트는 WWW에서의 실시간 공동 협업 환경을 목적으로 수행 되었으며[14], 이 프로젝트는 참가자나 각 팀이 웹 브라우저와 멀티미디어를 이용하여 정보를 공유함으로써 공동 협업을 가능하게 한다. 또한 MADEFAST 프로젝트는 WWW을 이용하여 다양한 분야에서 지리적으로 분산된 팀간의 협력 설계를 보여주는 성공적인 사례이다[15]. 스탠포드 대학교의 Next-Link 프로젝트에서는 분산설계와 동시공학을 위한 시스템에 협력 에이전트를 이용하였고[16], Erkes 등의 논문에서는 인터넷에서 생산 서비스를 공유하기 위한 실용적인 접근 방법으로써 HTTP와 CGI, Tcl을 사용하였다.

본 연구에서는 웹 기반 엔지니어링을 위한 여러 기술들 중 ObjectWeb을 이용하여 다중 사용자간 3차원 CAD 데이터를 기반으로 하는 컨퍼런스가 가능한 웹 기반 시스템을 구축하였다. 이 시스템의 클라이언트는 자바 애플릿으로써 사용자에게 제공되며 서버는 SDAI-Java에 의해 구축된 객체 지향 데이터베이스를 이용하여 STEP으로 기술된 제품 데이터에 접근한다. 객체간 통신을 위한 구조로 CORBA를, 다른 시스템과 데이터의 공유를 위해 STEP을 사용하고 응용 프로그램을 위한 언어로써 자바를 이용하였다.

본 논문의 구성은 다음과 같다. 2절에서는 네트워크를 기반으로 하는 엔지니어링 환경에 적용되는 기술들에 관해 간략하게 언급한다. 3절에서는 STEP 데이터를 영속화하여 객체지향 데이터베이스에 저장하고 접근하는 방법에 대해서 기술하고 4절에서는 구현된 데이터 베이스와 연계되어 운용되는 컨퍼런스 시스템의 구조 및 구현 방법에 대해서 논의한다.

2. 네트워크 기반 엔지니어링 요소 기술

WWW 환경에서 공동 작업이나 데이터의 공유 등 엔지니어링과 관련된 문제를 다양한 측면에서 적용하여 이용하기 위해서는 서버와 클라이언트간 원활한 데이터와 메시지의 교환이 가능하여야 한다. 이를 위한 기술로서 대표적이며 널리 사용되는 것으로는 CGI(Common Gateway Interface)와 CORBA-Java를 들 수 있다.

2.1 CGI와 CORBA, Java

CGI는 문서에 HTML 태그(tag)형태로 포함된 폼(form)을 유저 인터페이스로 하여 사용자의 입력을 파라미터로 서버 사이드의 스크립트나 실행파일을 실행시키고 그 결과로 HTML이나 특정 MIME 타입의 데이터를 클라이언트의 웹 브라우저에 보내주는 방식으로 동작한다[18]. 이러한 스크립트나 실행파일은 웹 서버의 자식 프로세서(child processor)로써 매번 독립된 인스탄스로 실행되기 때문에 많은 사용자의 빈번한 CGI사용은 서버에 부하를 주게 될 수 있고, 유저 인터페이스가 HTML의 폼으로 제한된다는 점과 한 번의 실행으로 한 번의 결과만이 전달되는 특징으로 서버와 클라이언트간 다양한 인터랙션이 요구되는 분야에 적용하는 데는 어려움이 있다[19].

CORBA는 OMG(Object Management Group)에서 분산 객체들간의 상호작용(Object Inter-operability)을 위하여 제안한 프로그램의 구조에 대한 표준 방안이며, 어떤 기종이나 운영체제, 프로그래밍 언어에 제한 받지 않고 정보와 자원을 교환하기 위한 것이고,

현재 분산 객체기술의 산업 표준이다[20,21]. CORBA객체들 간의 관계는 IDL로 기술되어 객체 상호간의 다양하고 계층적인 데이터와 메시지의 교환이 가능하다. 또한 은닉화, 상속성, 다형성 등의 객체 지향 패러다임을 지원함에 따라 한 번 작성된 객체는 다른 객체에 의해 참조 되거나 확장될 수 있고 객체간 플러그-앤-플레이가 가능해진다.

또한 자바는 근래, 웹 기반의 프로그래밍 언어로서 각광을 받고 있다. 자바는 컴파일된 바이너리 레벨에서 플랫폼간 호환이 가능한 언어이며, 네트워크를 통한 코드의 이동이 가능하면서도 안정적인 보안을 제공하는 특성으로 인하여 짧은 역사에도 불구하고 WWW 환경에서의 유용성이 어느 정도 검증되었다.

2.2 ObjectWeb

ObjectWeb은 CORBA와 자바를 WWW에 통합시킨 것으로, 넷스케이프사가 1997년 6월 자사의 서버와 웹브라우저에 ORB를 포함시키면서 공식적으로 태어났다. 이로 인하여 웹은 CORBA와 Java가 어울려 제공하는 다양한 분산 서비스들과 객체 지향 패러다임을 이용할 수 있게 되었고, 상대적으로 CORBA와 자바는 대중적으로 널리 알려진 WWW이라는 통일된 인터페이스를 갖게 되었다.

Fig. 1은 ObjectWeb에서 클라이언트 애플릿의 이동과, 클라이언트/서버 객체들 간의 상호관계를 나타낸다[22]. 클라이언트쪽의 실행 프로그램인 자바 애플릿은 HTML 문서 안에 참조의 형태로 기술되어 있으며, 이러한 문서를 다운로드 받은 웹 브라우저는 이어서 애플릿을 다운로드 받고, 이를 웹 브라우저내의 자바 가상 머신에 로드 한다. 이후 애플릿이 실행되며, 애플릿은 CORBA의 기본 프로토콜인 IIOP (Internet Inter-ORB Protocol)를 통하여 서버 객체들과 통신하게 된다. 서버는 CORBA에 의해 구현된 객체뿐 아니라 기타 여러 객체들이나 데이터베이스와 연결될 수 있다[23]. 본 연구에서는 동시 공학적 분산 엔지니어링을 위한 프로토타입으로 ObjectWeb을 이용한 3D 컨퍼런스 시스템을 구현하였다. 클라이언트 서버의 구조를 가지는 이 시스템은 웹 서버를 통하여 자바 애플릿으로 작성된 클라이언트 프로그램이 사용자측으로 전달되어 실행된다. 이후, 여러 사용자들이 발생시키는 메시지와 이벤트는 CORBA를 통하여 서버에 전달되고, 서버는 이러한 메시지를 또 다른 사용자들에게 전달함으로써 컨퍼런스를 가능하게 한다. 또한 서버는 데이터베이스나 기타 기능성 모듈 또는 객체들과 연결되어 사용자들의 요구에 응답하고, 때로는 능동적으로 사용자에게 데이터와 메시지를 전달하게 된다.

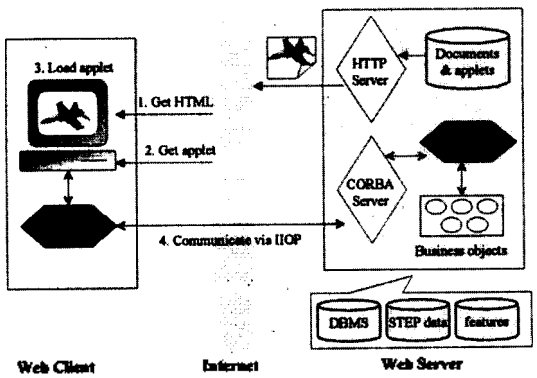


Fig. 1. Java, CORBA와 ObjectWeb.

3. STEP 데이터베이스 구축

STEP SDAI(Standard Data Access Interface)는 ISO 10303-22(STEP part 22)에서 언어에 독립적인 형태로 기술되어 있으며, Java에 대한 SDAI 바인딩은 파트 27로 제정 작업 중에 있는데, 네트워크의 부하를 고려하여 light SDAI와 heavy SDAI의 두 가지 방법에 대해 논의되고 있다[24].

STEP 표준을 CORBA 환경에서 적용하기 위한 SDAI-IDL에 대해서는 파트 26에서 정의하지만[25] 본 시스템에서 서버와 클라이언트간 정보의 교환이 STEP 스키마의 모든 엔티티를 다루는 것이 아니라 형상 및 비형상 정보와 컨퍼런스 관련 데이터에 한정되므로 IDL 레벨에서 SDAI를 사용하지는 않고 모델의 교환 및 컨퍼런스 관련 객체와 메쏘드를 정의하여 사용한다. 이의 일부를 appendix에 예시하였다.

서버쪽에서는 light Java-SDAI를 사용하여 구축된 객체 지향 데이터베이스를 사용하는데, 이로써 얻을 수 있는 이점은 모델이 데이터가 요구될 때 물리 파일을 파싱하고 인스탄스화 하는 과정을 거치지 않고 데이터 베이스에서 영속적인 객체를 바로 얻어냄으로써 서버의 부하를 줄이고, 데이터베이스에서 클라이언트의 요구에 대해 최소 객체 단위의 잠금(lock)을 행함으로써멀티쓰레드된 접근을 부드럽게 할 수 있어 공동 협업 모델링으로의 확장 시 순조로운 서버 객체 접근을 기대할 수 있다는 점이다.

Fig. 2는 EXPRESS 스키마와 STEP 물리 파일로부터 자바 기반의 객체 지향 데이터베이스를 구축하는 과정을 보여준다.

가장 먼저, ST-Developer 1.6에 포함된 EXPRESS-

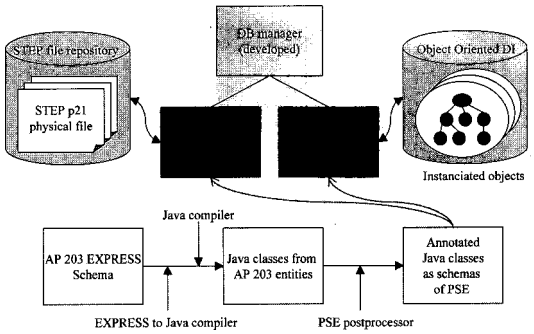


Fig. 2. Process of developing DB using SDAI-Java.

to-Java 컴파일러를 이용하여 EXPRESS 언어로 기술된 스키마로부터 각각의 엔티티에 해당하는 자바 클래스들의 소스 파일을 생성하는데, 이 클래스들은 물론 SDAI-Java의 바인딩 규칙을 따른다. 이렇게 생성된 자바 소스 파일을 컴파일하여 바이트 코드로 된 자바 클래스 파일들을 생성하면 응용 프로그램에서 이들 클래스를 사용할 수 있다. 본 연구에서 객체 지향 데이터베이스 엔진으로 사용하는 Object Design 사의 PSE Pro for Java에서는 자바 클래스 자체를 데이터베이스의 스키마로 사용하는데, 그냥 쓰지는 않고 몇 가지 영속화를 위한 메쏘드를 추가하여야 하는데, 이 과정에서 PSE에 포함된 클래스 후처리기를 사용하여 바이너리 레벨에서 메쏘드를 추가한다.

이렇게 새로운 메쏘드가 추가된 클래스는 ST-Developer의 JavaROSE 또는 세션 레벨 바인딩에 의해 여전히 접근 가능할 뿐 아니라 PSE에 의해 데이터베이스에 저장될 수 있다[26,27].

데이터 베이스 관리자는 ST-Developer의 Java API를 이용하여 STEP 파일 저장소로부터 STEP 물리 파일을 읽어 들이고 데이터를 인스탄스화한다. 이렇게 인스탄스화된 객체들은 영속화 엔진에 의해 그 상관관계를 유지하면서 영속화 됨으로써 데이터베이스에 저장된다. 물론 앞서 설명한대로 이 데이터베이스의 스키마는 SDAI-Java 바인딩 규칙에 의한 EXPRESS 스키마의 자바 버전이며, 영속화 엔진이 필요로하는 몇 가지 메쏘드가 추가되어 있는 자바 클래스 자체이다. 컨퍼런스 시스템의 서버는 사용자의 요구에 따라 데이터베이스와 영속적 객체들에 접근하고, ORB를 통하여 이들을 클라이언트에 제공하게 된다.

4. 3D CAD데이터 원격참조 및 컨퍼런스

4.1 시스템의 구조 및 구현 환경

CORBA를 이용한 분산 시스템의 구현에 있어서 첫번째 단계는 IDL 언어로써 분산 객체간 인터페이스를 정의하는 단계이다. IDL로 기술된 인터페이스는 IDL 컴파일러에 의해 시스템 구축에 사용될 프로그래밍 언어로 컴파일되어 서버 및 클라이언트에서 사용된다. IDL의 컴파일에 의해 생성되는 코드는 각각 서버 스켈리턴과(skeleton) 클라이언트 스텁(stub)으로써, CORBA와 구현 언어간을 연결하는 역할을 한다. 본 시스템에서 클라이언트의 전체 모듈과 서버의 대부분 모듈은 자바로 작성되었고 서버 객체 구현부에서는 JNI(Java Native Interface)를 통하여 이미 C++로 작성된 솔리드 모델 및 곡면의 가시화를 위한 모듈을 사용한다.

Fig. 3은 본 시스템의 전체적인 구조를 나타내고 있다. 크게 클라이언트-서버의 구조를 가지며 서버와 클라이언트는 CORBA를 통해 상호 통신하고 있다.

서버는 컨퍼런스를 위해서 여러 개의 세션을 관리할 수 있어야 하고, 세션은 각각의 클라이언트에 능동적으로 데이터를 보낼 수 있어야 하기 때문에 세션에 참가한 클라이언트의 참조 객체를 리스트로 가지고, 마크업이나 채팅 텍스트 등 사용자가 발생시키는 데이터를 관리하여 클라이언트들에 분배하기 위한 관리자를 가진다. 서버 구현부의 오른쪽에 나타난 여러 개의 conference session들은 이러한 기능을 위한 동적 생성 및 소거가 가능한 세션들이며 왼쪽의 모듈들은 세션에 의해 사용되는 서버의 주요 부분을 나타내고 있다.

Conference event manager는 컨퍼런스 시스템에 접속된 여러 클라이언트의 사용자 메시지를 받아들이고 이를 다른 사용자들에게 전송하거나, 현재 진행되고 있는 관점과 의견을 보관하여 중간에 참여하게 되는 참가자에게 전달하는 역할을 한다. 이러한 기능의 대상이 되는 데이터는 여러 사용자들이 지정한 마크업이나 채팅, 뷰 데이터 등이다.

솔리드 모델 또는 그 외의 형상 데이터는 경계만을 표현한 와이어프레임이나 모든 곡면을 표현하는 세이딩된 모델로 가시화 될 수 있는데, Visualization Module은 경계 표현의 다양체 모델(mainfold_solid_brep)로부터 곡면의 형상과 경계 곡선에 근거해 삼각화된 데이터를 생성하거나, 솔리드 또는 와이어프레임 모듈로부터 가시화를 위한 와이어 프레임을 추출하는 기능을 한다. 이 모듈은 C++로 작성되어 있으며 JNI(Java Native Interface)를 통하여 객체 구현부와 연결된다.

STEP entity navigator는 STEP 데이터가 표현하는 데이터를 엔티티 단위까지 접근 가능하게 하는 모듈로써, 각각의 객체는 클라이언트에서 객체를 접근할

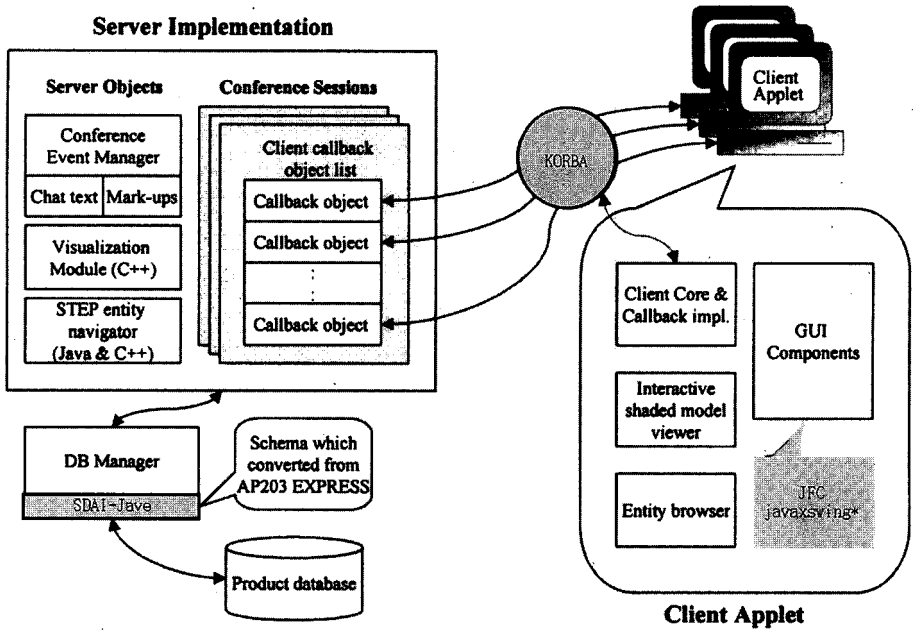


Fig. 3. 3D-conferencing system architecture.

수 있게 하기 위해 IDL로 정의되고, 그 구현 부에서 JNI를 통해 기존 C++로 작성된 모듈을 사용한다.

서버의 아래쪽에 나타나있는 데이터베이스는 제 3 절에서 기술한 내용에 의해 생성된 객체 지향 데이터베이스이며 이 것의 스키마는 SDAI-Java에 의해 자바로 매핑된 AP203 스키마 그 자체이며 데이터베이스에 저장된 객체들은 SDAI-Java 클래스들의 영속적 인스탄스이다. 데이터베이스 관리자는 서버로부터의 질의에 의해 적절한 객체를 검색하고 반환하는 기능을 한다.

클라이언트는 단순히 웹 브라우저만으로 실행 시킬 수 있어야 하기 때문에 순수 자바로 작성된다. Client core는 IDL로부터 생성된 스텁 코드를 확장하여 서버의 서비스를 사용할 수 있도록 하는 핵심 부이고, client callback은 콜백 객체의 구현부이며, 서버가 호출하고 클라이언트가 반응 할 수 있게 하는 부분이다. 예를 들어 서버가 한 사용자에게 뷰 변경 메시지를 받으면 다른 사용자들의 클라이언트측의 콜백 메쏘드를 호출하게 되고, 그 구현부에서는 랜더러에 뷰 정보를 전달함으로써 화면이 갱신되게 한다. 이로써 모든 사용자가 같은 뷰를 공유할 수 있게 된다.

뷰어는 와이어 프레임을 보여 주거나 삼각형 데이터를 받아 세이딩 해서 보여주는 기능을 한다. 또한 마우스 메시지에 반응하여 화면을 확대, 축소하거나 회전, 이동이 가능하며 이러한 뷰 변경 메시지를 컨퍼런스 서버에 전달하고, 전달 받는 기능을 가지는데, 뷰 변경시 화면의 크기나 종횡비가 다른 사용자들 간에도 일관된 뷰를 공유할 수 있어야 하므로 정규화된 뷰포트 데이터와 회전 변환 행렬을 전송하거나 받게 된다. 또한 뷰어는 마크 업 데이터나 엔티티 브라우저에 의한 선택 부분에 해당하는 형상을 별도로 표시하는 기능 등을 가진다. 이 뷰어의 구현을 위해서 Java3D의 사용이 고려되었지만 현재 Java3D는 OpenGL 기반으로, 플랫폼에 종속적인 동적 라이브러리를 사용하기 때문에 최소한의 랜더링을 위한 모듈을 순수 자바 언어로 구현하였다.

엔티티 브라우저는 STEP 데이터를 이루고 있는 엔티티들의 포함관계를 볼 수 있는 트리 컨트롤을 제공한다. 트리의 각 노드들은 서버 객체와 연결되어 있으며, 형상 엔티티의 경우 뷰어에서 그리고 있는 형상과도 연결된다. 트리 컨트롤에서 항목을 클릭했을 때, 서버에 그 하부 항목에 대한 질의를 행하고 데이터를 얻어 다시 하부 항목으로 확장할 준비를 한다. 즉, 엔티티 브라우저는 트리의 모든 항목을 서버로부터 받아와서 보여 주는 것이 아니라 필요에 따라 사용자가 브라우징 하려하는 항목만 실시간으로 얻어낸다. 이 방법은 데이터의 이동 양과 클라이언트측의 메모리 소요를 크게 줄인다.

대화 상자나 채팅 창 등의 유저 인터페이스는 JFC (Java Foundation Classes, swing)를 사용하고 있다.

JFC는 자바 1.1 까지는 별도의 패키지(com.sun.java.swing)로 제공되었지만 자바 1.2 부터는 기본 패키지(javax.swing)로 제공되고 있다.

서버와 클라이언트를 작성하는 도구로써, IDL 컴파일러와 ORB로는 IONA Technologies의 OrbixWeb 2.0을 사용하고, STEP 데이터를 다루기 위해 Rose library와 SDAI-Java를 위한 도구 및 API를 제공하는 STEP Tools의 ST-Developer 1.6을, 객체 지향 데이터베이스로 Object Design사의 PSE Pro for Java 2.0을 사용하고 있다.

4.2 컨퍼런스 세션과 콜백 메커니즘

Fig. 3에 나타나 있는 바와 같이 세션은 세션에 참여한 클라이언트들을 참조하고 있으며 클라이언트는 자신이 참여한 세션을 참조하고 있다. 클라이언트는 세션 객체에 데이터나 질의 등의 서비스를 요구할 수 있고, 세션은 콜백 매커니즘을 통하여 클라이언트에 데이터를 전달한다. 예를 들어서, 세션의 참여자가 뷰를 변경하였다면 클라이언트는 변경된 뷰의 데이터를 세션 객체에 전달하고 세션은 참여한 다른 클라이언트에게 변경된 뷰 데이터를 분배하여 줌으로써 세션 참여자들이 동일한 뷰를 공유할 수 있게 된다.

Appendix 1은 IDL에서의 세션의 정의를 나타내고 있다. 정의된 메쏘드는 크게 컨퍼런스를 위한 메쏘드와 형상 및 비형상 정보를 제공하기 위한 메쏘드로 나눌 수 있다.

단방향(oneway)으로 정의된 메쏘드들은 컨퍼런스에 관련된 것들로서, 클라이언트-서버간의 고착(deadlock)을 피하기 위하여 단방향으로 정의되었다. 사용자의 뷰가 변경되었을 때 서버에 알리는 updateViewingMatrix()와 setOrtho(), 마크업과 채팅 메시지의 전달을 위한 setMarkUp(), postMessage()와 같은 메쏘드가 컨퍼런스 관련 메쏘드에 해당한다. 또한 모델의 삼각화된 데이터를 받는 것과 같이 데이터의 이동에 시간이 걸릴 수 있는 기능은 일단 단방향 메쏘드로 클라이언트가 요구하고 서버는 별도의 쓰레드를 생성하여 데이터를 전달함으로써 고착상태를 방지한다. 따라서 getTriangleData()는 클라이언트가 서버로부터 데이터를 얻기 위하여 호출을 하지만 반환 되는 값이 없고, 호출되면 서버의 데이터 전달 쓰레드가 작동하여 클라이언트의 콜백 메쏘드를 통해 별도로 데이터를 전달하게 된다.

세션의 정의에 나타나 있는 SCObject는 서버측에 활성화된 모델이 포함하는 객체 단위의 정보에 접근하기 위한 인터페이스로서, 임의 객체에 대한 접근 방법을 제공한다. 이 객체의 정의는 추후 IDL에 SDAI-IDL을 적용하게 되면 SDAI_DAObject 와 같은 표준 객체로 대체될 것이다[25].

Appendix 2는 클라이언트 콜백을 위한 IDL 정의를 보여주고 있다. 콜백은 일반적인 CORBA 객체들과는 반대로 클라이언트에 콜백 객체의 구현부가 있고, 서버 측에서 호출을 할 수 있게 되는 메커니즘을 가진다. 이를 이용하여 서버에서 능동적으로 클라이언트에 데이터나 메시지를 전달 할 수 있다. Appendix 2에 나타난 메쏘드들은 세션의 정의와 마찬가지로 컨퍼런스 관련과 모델 정보 관련의 두 부류로 되어 있으며, isAlive()를 제외한 모든 메쏘드가 단방향이다. 컨퍼런스 관련 메쏘드는 채팅 메시지를 전달하는 newMessgae(), updateViewingMatrix(), setOrtho(), setMarkUps() 등이며, 그 외의 메쏘드 들은 주로 서버 측에서 멀티 쓰레드를 이용하여 클라이언트에 데이터를 전달하는데 사용된다. 이러한 메쏘드들은 클라이언트로부터 데이터와 메시지의 전달 성공 여부를 확인하지 않지만 서버의 클라이언트 관리 쓰레드는 주기적으로 isAlive()를 호출함으로써 클라이언트가 제대로 동작하고 있는지를 판단할 수 있다.

4.3 작업 시나리오

본 연구를 통해 개발된 컨퍼런스 시스템은 Object-Web 기술에 기반을 두고 있으며 클라이언트-서버 구조로 되어 있다. 분산된 다수의 사용자는 클라이언트 프로그램을 통해 모델과뷰, 마크업 데이터 등을 공유하고 서로의 의견을 교환 할 수 있다.

서버는 SDAI-Java mapping에 의해 자바 클래스로 변환된 객체들이 저장되어 있는 데이터베이스로부터 모델을 검색하여 읽어 들이고, 이에 대한 형상 및 비형상 정보를 계층적으로 보여주기 위한 모듈과 솔리드 모델을 삼각화하여 세이딩된 형상으로 보여주기 위한 가시화 모듈 등 데이터 공유를 위한 객체들과, 사용자간 대화나 뷰, 마크업 데이터 등을 관리하여 클라이언트에 배분하여 주고 세션을 관리하는 컨퍼런스 관련 객체들로 구성되어있으며 모델 정보에 대한 객체지향 데이터베이스와 함께 구동 된다.

서버는 클라이언트의 요구에 따라 세션을 만들거나 사용자를 세션에 참가 시키며, 한 사용자가 발생시키는 메시지를 다른 클라이언트에 중계한다. 이때, 뷰의 변경과 같은 메시지는 매우 빈번히 발생하고, 클라이언트측의 랜더링 성능에 따라 반응 속도가 달라지며, 각각의 클라이언트에서 상이한 메시지를 동시에 보낼 수도 있기 때문에 서버는 전달되는 메시지를 발생시

켜 클라이언트에 단방향으로 전달함으로써 속도와 데이터의 일관성을 유지한다. 그러나 서버가 네트웍 트래픽이나 정전, 기타 예상치 못한 오류로 죽거나 접속이 끊어지는 클라이언트를 체크하는 별도의 쓰레드를 실행시키기 때문에 이러한 일방통행 방식이 데이터 전달에 대한 신뢰성이나 안정성에 문제를 미치지는 않는다.

클라이언트는 서버와의 연결부, 세이딩 모델 및 와이어프레임 모델에 대해 마우스에 의한 줌, 팬, 회전 등의 기능을 제공하는 랜더링 모듈, 형상 및 비 형상 데이터의 검색을 위한 엔티티 브라우저, 마크업 유저 인터페이스, 채팅 윈도우 등으로 구성되어 있다.

클라이언트 프로그램은 자바 애플릿으로 되어 있기 때문에 사용자가 애플릿을 포함하고 있는 웹 페이지를 웹 브라우저로 접근하면 사용자의 컴퓨터로 다운로드 되어 실행된다. 이후 사용자는 컨퍼런스 서비스를 제공하는 서버와 사용자의 이름을 입력하여 서버에 로그인 하고, 새로운 세션을 만들거나 기존의 세션에 참여 할 수 있게 된다. 기존의 세션에서 특정 모델에 대한 컨퍼런스가 진행 중이라면 참여하기 전 세션의 개설자와 모델의 이름을 알 수 있다. Fig. 4는 세션에 참여하기 위한 대화상자를 보여준다.

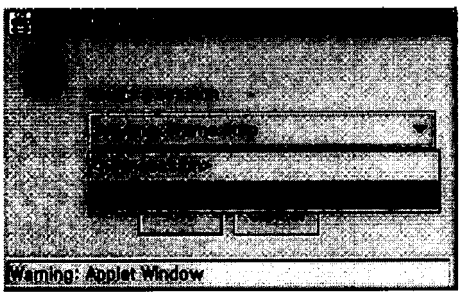


Fig. 4. Joining a session.

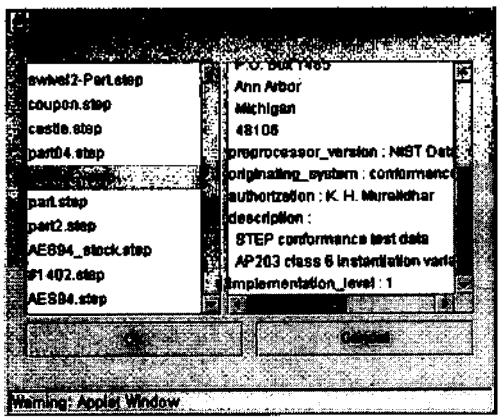


Fig. 5. Data selection dialog of client applet.

Fig. 5는 리스트로부터 대상 모델을 선택하는 화면이다. 왼쪽 창은 서버의 데이터베이스에 저장된 STEP 모델의 리스트이며, 오른쪽 창은 선택된 모델의 file_name과 file_description 엔티티들의 내용을 보여주는 일종의 미리 보기(preview)이다[28]. 사용자가 왼쪽 창에서 데이터를 클릭하면 클라이언트는 데이터 이름을 키 값으로 하여 ORB를 통해 서버에 질의하여 오른쪽에 그 내용을 보여주게 된다.

Fig. 6은 실행된 클라이언트 애플릿의 예를 보여주고 있다. 세이딩된 모델과 함께 오른쪽 일부가 가려진 창에 이와는 관점이 다른 다른 와이어 프레임 모델이 보이고 엔티티 브라우저에서 선택된 페이스와 마크업된 꼭지점이 나타나있다. 또한 채팅창을 통해 세션에 연결된 다른 사용자와 의견을 주고 받는 것을 볼 수 있다. 전체적인 외관은 프레임을 생성해서 클라이언트를 실행하기 때문에 웹 문서에 포함된 형태는 아니다.

사용자는 모델이 나타나 있는 창에서 마우스를 드래그 함으로써 모델의 확대/축소, 회전, 이동 등 뷰 변경을 할 수 있으며 모델 창 위의 버튼을 통해 자신이 보고 있는 뷰를 다른 참가자 들과 공유 할 것인지 안할 것인지를 토글할 수 있다. 또한 별도의 뷰를 위해 하나의 모델에 대한 여러 개의 창을 사용할 수 있으며, 세이딩 여부를 결정할 수 있다. 디폴트는 와이어 프레임을 보여주지만 세이딩 버튼을 누르게 되면 서버에 삼각형 데이터를 요구하게 되고 서버는 솔리드 모델의 가시화 과정을 거쳐 삼각형 데이터로 만들어 보내주게 된다. 삼각형의 생성과 전송시에도 컨퍼런스는 지장 없이 진행 되어야 하므로 서버는 이 일을 멀티쓰레드를 통해 해결 한다.

화면의 오른쪽 위에 보이는 것은 엔티티 브라우저로써 클라이언트 측에서 서버측 객체들의 포함관계를

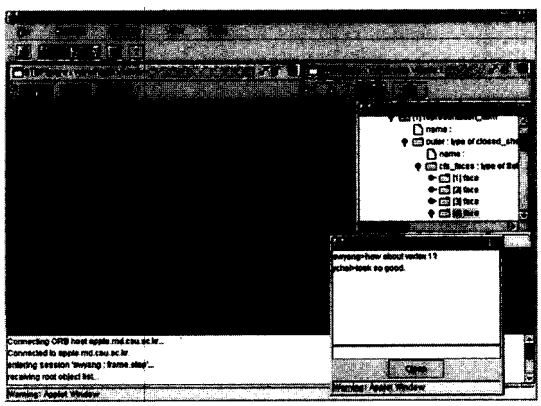


Fig. 6. An example of running client applet.

계층적으로 보고 텍스트나 수치 값을 확인할 수 있다. 또한 곡면이나 에지 데이터를 선택하면 모델 뷰에서 해당 부분을 나타내어 준다.

화면의 오른쪽 아래 창은 채팅 창으로써 사용자간 의견 교환을 할 수 있게 해준다. 채팅 창이 활성화 되어 있지 않은 상태에서 채팅 메시지를 받게 되면 클라이언트는 메시지 통보와 함께 창을 활성화 시킨다.

5. 결 론

본 연구에서는 CORBA-Java와 웹, 그리고 SDAI-Java에 의한 객체지향 데이터베이스가 연동하는 3D 컨퍼런스 시스템을 구현하였다. 클라이언트를 Java로 작성함으로써 사용자는 웹 브라우저만으로 시스템을 사용할 수 있게 되었고, 최소 객체 단위의 인터페이스로 네트워크의 부하를 줄이고, 다중 사용자의 데이터베이스로의 접근을 원활하게 하는 시도를 하였다.

현재 가장 문제가 되는 것은 클라이언트 애플릿의 크기인데, 클라이언트 모듈 전체의 크기는 100kB 정도이지만 GUI를 위한 패키지와 ORB를 위한 패키지의 크기가 2MB 가까이 되어 웹상에서 실시간으로 필요한 모듈 전체를 다운로드 받는 데는 문제가 있다. 추후 웹브라우저가 Java 2(공식적으로 Java 1.2)를 지원하게 되면 100kB 정도의 애플릿 패키지만이 웹을 통해 이동하게 되므로 애플릿 다운로드에 따른 네트워 부하와 시간이라는 실용적 문제는 해결될 수 있다.

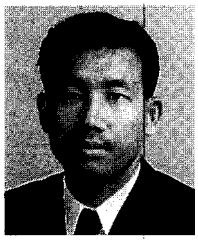
구현된 시스템의 구조는 사용자 메시지에 대해서는 양방향으로, 제품 모델 정보에 대해서는 단방향의 데이터 흐름을 허용하지만, 모델 정보에 대한 양방향 교신을 가능하도록 확장한다면 전체적인 구조의 변경 없이 분산 환경하에서의 협업 모델링에 적용이 가능하도록 고려하였다.

현재 구현된 클라이언트와 서버의 인터페이스는 컨퍼런스를 위한 최소한 객체와 그 메쏘드들로 정의되어 있다. 하지만 앞으로는 CORBA와 SDAI에 기반하여 구현된 다른 분산 객체 시스템과 연계하여 구동할 수 있도록 SDAI-IDL에 의한 인터페이스로 대체되어야 한다. 현재 light weight SDAI-Java에 의한 데이터베이스 부분도 추후 full SDAI의 기능을 완전히 제공할 수 있도록 확장되어야 할 것이다. 물론 이러한 연구는 SDAI-IDL과 SDAI-Java의 표준 제정 작업 성과와 진도에 맞추어 진행되어야 할 것이다.

참고문헌

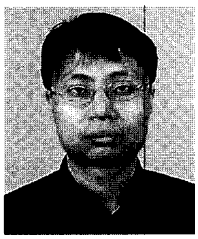
1. Regli, W.C., "Internet-Enabled Computer-Aided Design", *IEEE Internet Computing*, Vol. 1, No. 1, pp. 39-50, 1997.
2. Potter, C., "Web-Enabled Engineering: step-by-step", *Computer Graphics World*, pp. 64-69, Nov., 1997.
3. 김철영, 김남국, 김영호, 강석호, "웹과 STEP을 이용한 제품 설계 공유 정보 시스템", *한국 CAD/CAM학회 논문집*, Vol. 1, No. 3, pp. 203-214, 1996. 12월.
4. Huang, J.Y., Fang-Tsou, C., T., and Chang, J., L., "A Multiuser 3D Web Browsing System", *IEEE Internet Computing*, Vol. 2, No. 5, pp. 70-79, 1998.
5. Choi, Y. and Shin, H.Y, "Sharing STEP data on the Internet", *Proceedings of the 8th International Conference on Production Engineering*, Sappro, Japan, Aug. 18-20, pp. 397-405, 1997.
6. Hardwick, M., Spooner, D.L., Ranho T. and Morris K.C., "Sharing manufacturing information in virtual enterprises", *Communications of the ACM*, Vol. 39, No. 2, pp. 46-52, 1996.
7. Pahng, H., Senin, N. and Wallace, D., "Distribution modeling and evaluation of product design problems", *Computer-Aided Design*, Vol. 30, No. 6, pp. 411-424, 1998.
8. Martino, T.D., Falcidieno, B., and Haβinger, S., "Design and engineering process integration through a multiple view intermediate modeler in a distributed object-oriented system environment", *Computer-Aided Design*, Vol. 30, No.6, pp. 437-452, 1998.
9. Han, J.H. and Requicha, A.A.G., "Modeler-independent feature recognition in a distributed environment", *Computer-Aided Design*, Vol. 30, No. 6, pp. 473-486, 1998.
10. Gadh, R. and Sonthi, R., "Geometric shape abstractions for internet-based prototyping", *Computer-Aided Design*, 1998, Vol. 30, No. 6, pp. 473-486, 1998.
11. Dong, A. and Agogino, A., M., "Managing design information in enterprise-wide CAD using smart drawings", *Computer-Aided Design*, Vol. 30, No. 6, pp. 425-435, 1998.
12. Debras, P., Huguet, P., Amar, V. and Poyet, P., "RISESTEP (EP 20459): a user driven project to develop a platform for product data sharing in a concurrent engineering context", *Proceeding of the Product Data Technology Days 1997*, pp. 23-30, French Rivera, France, 1997.
13. Marache, M., Amar, V. and Diard, F., "A CORBA-based infrastructure managing STEP distributed models for virtual reality applications", *Proceeding of the Product Data Technology Days 1997*, pp. 119-128. French Rivera, France, 1997.
14. Charles, P., "SHARE: A Methodology and Environment

for Collaborative Product Development", http://cdr.stanford.edu/SHARE/share.html.
15. Cutkosky, M.R., Tenenbaum, J.M. and Glicksman, J., "Madefast: Collaborative engineering over the Internet", *Communications of the ACM*, Vol. 39, No. 9, pp. 78-87, 1996.
16. Charles, P., "NEXT-LINK", http://cdr.stanford.edu/NextLink/NextLink.html.
17. Erkes, J.W., Kenny, K.B., Lewis, J.W., Sarachan, B.D., Sobolewski, M.W. and Sum, R.N., Jr., "Implementing shared manufacturing services on the World-Wide Web", *Commnunications of the ACM*, Vol. 39, No. 2, pp. 34-45, 1996.
18. Robinson, D.R.T., "The WWW Common Gateway Interface Version 1.1", IEFT, Feb. 1996
19. Evans, E. and Rogers, D., "Using Java Applets And CORBA for Multi-User Distributed Applications", *IEEE Internet Computing*, Vol. 1, No. 3, pp. 43-55, 1997.
20. OMG, *The Common Object Request Broker Architecture and Specification*, Aug, 1996.
21. Siegel, J., *CORBA fundamentals and programming*, John Wiley & Sons, Inc., 1996.
22. Orfali, R., Harkey, D., and Edwards, J., "CORBA, Java, and the Object Web", *Byte magazine*, http://www.byte.com/art/9710/sec6/ art3.htm, Oct., 1997.
23. Vogel, A. and Duddy, K., *Java Programming with CORBA*, John Wiley & Sons, Inc., 1997.
24. Sellentin, J., "Mapping the SDAI of STEP to Java", http://phrames.informatik.tu-muenchen.de/javasdai/, 1997.
25. TC184/SC4, *Part 26: Interface Definition Language Binding to the Standard Data Access Interface Specification, Working Draft*, May 30, 1996.
26. Object Designs, *Object Store PSE pro for Java users manual*, 1998.
27. STEP Tools Inc, *ST-Developer v1.6 Online Manuals*, 1997, http://www.steptools.com/support /stdev_docs/.
28. ISO, *ISO 10303-21 - Part 21: Implementation methods: Clear text encoding of the exchange structure*, 1994.



최 영

1979년 서울대학교 기계설계학과 학사
1981년 한국과학원 생산공학과 석사
1989년 Carnegie Mellon University 기계공학과 박사
1981년~1984년 대우중공업 중앙연구소 연구원
1989년~1990년 Engineering Design Research Center 연구원
1992년~현재 중앙대학교 기계설계학과 교수
관심분야: 네트워크 CAD, Non-manifold modeling, 솔리드모델링, 곡선 및 모델링



양 상 욱

1998년 중앙대학교 기계설계학과 학사
1998년~현재 중앙대학교 기계설계학과 석사 과정
관심분야: network-enabled CAD, solid modeling

부 록

Appendix 1. Session의 IDL 정의 (발췌)

```
interface SCSession {
  oneway void updateViewingMatrix (
       in floatList matrix, in string userName);
  oneway void setOrtho(in float minx, in float maxx,
       in float miny, in float maxy,
       in string userName);
  oneway void setMarkUp(in float x, in float y,
    in float z, in float r, in string text, in string
    userName);
  oneway void postMessage(in string message, in
    string userName);
  oneway void selectEdges(in longList edgeOidList,
    in string userName);
  oneway void cancelSelectEdges();
  oneway void cancelMarkUps();
  oneway void deleteMarkUp(in int idx);

  MarkUpList getMarkUps() raises (RCIException);
  longList   getSelectedEdgeOids()
       raises (RCIException);

  string getOwner();
  string getModelName();

  stringList getDataFileList();
  string getPreview(in string design)
       raises (RCIException);
```

```
void useDesign(in string design,
        in string userName) raises (RCIException);

SCObject getObject(in long oid);
SCObjectList getRootObjects()
        raises (RCIException);
SCPolylineData getPolylineData()
        raises (RCIException);

floatList getViewingMatrix()
        raises (RCIException);
floatList getOrthoData() raises(RCIException);

oneway void getTriangleData(in string userName)
        raises (RCIException);
};
```

Appendix 2. Client callback의 IDL 정의 (발췌)

```
Interface SCCallback {
oneway void newMessage(in string message);
oneway void updateViewingMatrix(in floatList matrix);
oneway void setOrtho(in float minx, in float maxx,
        in float miny, in float maxy);

oneway void setRootObjects(
        in SCObjectList objList);
oneway void setPolylineData(
        in SCPolylineData polylines);
oneway void setStatus(in string status);

oneway void setMarkUps();
oneway void cancelMarkUps();
oneway void setSelectedEdge();
oneway void cancelSelectEdges();
/*methods for triangle data*/
oneway void beginTriangleTransfer();
oneway void setNumberOfSurface(
        in long number);
oneway void setSurfaceData(
        in floatList TrianglePts, in long objectID);
oneway void endTriangleTransfer();
boolean isAlive();
};
```