□ 기술해설 □

분산 개발 환경에서의 객체지향 시각 프로그래밍

경북대학교 김상욱*
시스템공학연구소 진윤숙*

● 목 차 ●

1. 서 론

초고속 정보 통신망의 발달로 인한 기존의 소프트웨어의 개발 환경이 급속히 변화되었다. 이러한 변화 중 분산 개발 환경에서의 소프트웨어 개발 환경은 초고속 정보통신망의 사용을 촉진시키는 역할을 한다. 이러한 개발 환경 중에서 개발 프로세스를 지원하는 소프트웨어와 서비스를 요구하는 지원환경과 그룹웨어 등을 효과적으로 사용할 수 있다. 개발 환경과 서비스 개발 환경을 효과적으로 지원하는 도구로 시각 프로그래밍을 사용할 수 있는데, 이는 초고속 정보 통신망의 분산 개발 환경에서 효과적으로 사용될 수 있는 도구이다. 네트워크와 인터넷의 사용이 증가되고 있기 때문에, 분산 시스템이 넓은 범위의 분산 정보를 효율적이며 융통성 있도록 자원을 관리하여야 한다.

분산 컴퓨팅 기본 원리 중 하나는 어떤 처리를 국지적으로 처리되지 못할 경우에는 임의의 다른 지역에 위치하고 있는 컴퓨터에서 처리되도록 하여야 한다. 이러한 분산 컴퓨팅은 분산 객체 관리 구조(Distributed Object Management Architecture : DOMA)를 사용하는 분산 컴퓨팅 모델이 되어야 한다. 그러므로 객체지향 시각 프로그래밍 시스템은 분산 개발 환경을 이루고 있는 통신 프로세스를 기반으로 하여 구축되어야 한다.

기존의 여러 시스템이 분산 환경을 지원하더라도, 객체지향 시각 프로그래밍의 관점에서의 분산 개발 환경을 지원하는 시스템은 많지 않으며, 분산 모델과 객체지향 시각 프로그래밍의 융통성을 가지는 시스템도 별로 없다. 기존 시스템들은 대부분 클라이언트/서버형 모델이기 때문에, 사용자가 분산 자원으로부터 분산 컴퓨팅 환경을 형성하려면 주소지정 자료구조 정의 등의 많은 제한을 가지게된다[5].

World Wide Web(WWW)[2]과 같은 분산 환경에서의 시스템들은 제공되는 플랫폼의 관

*정회원

점이나 프로토콜의 관점에서 분산 개발 환경에서의 객체지향 시각 프로그래밍 모델을 지원하지 않는다. 뿐만 아니라 분산 처리 능력 면에서도 융통성이 없다. WWW은 Hypertext Mark-up Language(HTML)[3]라는 언어를 사용하여 도큐먼트들의 링크를 형성하고 있지만, 이 HTML 언어는 이미지나 동영상 같은 다른 포맷으로 저장된 도큐먼트에서는 네비게이션할 수 없다. 링크 관리 서브시스템이 부족하여 서버에서 서버로 연속적으로 이동되도록 되어있다.

Hyper-G 시스템[1]은 WWW와 비슷하지만, 링크 정보를 분리하여 링크 데이터베이스 내에 저장한다는 점에서 좀 더 앞선 개념이다. 이 기술은 이미지나 텍스트 도큐먼트 등 어떤 종류의 도큐먼트 형식이라도 링크를 적용하게 한다. 그러나 Hyper-G 서버는 간단한 시각 프로그래밍 모델이며 확장기능은 지원하지 않는다.

Hyperform[14]은 확장 가능한 시각 프로그래밍 서버이다. 이 기술은 특별한 객체의 시각 프로그래밍 서비스를 위한 객체 클래스를 제공하고, 시스템 내로 분산 객체들을 통합할 수 있다. 이 통합 기술로 Hyperform 서버 사이의 상호 운용이 가능하지만 완전하게 지원되지는 않는다.

Virtual Notebook System(VNS)[4,12]은 루틴의 데이터베이스를 기반으로 한 분산 시각 프로그래밍 구조이다. 이 시스템은 다른 VNS 서버 사이에서 상호 운용이 가능하며, 각 작업 그룹 사이에서 정보를 공유할 수 있기 때문에 커스텀 프로그램을 개발할 수 있다.

COSP[24]는 분산된 워크스테이션 환경에서 객체지향 시각 프로그래밍 기능을 지원하지만 객체의 투명성을 보증하지 못하고 각 사용자는 객체의 위치와 주소를 알아야만 시각 프로그래밍을 할 수 있다.

이러한 시스템들은 분산 환경을 지원하지만 그 분산 정도는 미약하다. "분산 개발 환경에서의 객체지향 시각 프로그래밍" 모델의 목적은 객체지향 시각 프로그래밍 기능 중에서 각 객체의 이벤트에 대한 오퍼레이션의 분산성을 허용하고, 사용자에게는 최대의 융통성을 제공하는 시스템을 제안하는 것이다. 이러한 객체지향 시각 프로그래밍의 분산 개발 환경으로의 확장성은 객체지향 시각 프로그래밍의 여러 객체의 기능을 다중 사용자들이 공유하도록 한다.

객체지향 시각 프로그래밍은 객체지향 기술과 시각 프로그래밍 기술을 결합한 프로그래밍 기술이다. 객체를 시각적으로 결합하였으며, 분산된 객체를 분산 개발 환경에서 효율적인 관리를 통하여 분산 개발 환경의 훌륭한 도구로 사용할 수 있다. 이 도구는 객체지향 프로그래밍 언어를 시각 구문을 가지도록 하거나 텍스트 지향 언어로 쓰여진 프로그램을 연결 사용하는 그래픽 도구환경이다.

객체지향 시각 프로그래밍 서비스를 요구하는 객체 단위의 프로그래밍 기능을 다른 위치에 있는 여러 컴퓨터 사용자에게 지원하여야 하는데, 이러한 목적은 사용자가 원하는 모든 객체에 공통 능력을 지원하여 시스템 기능의 확장성이나 융통성을 최대화한다. 객체지향 시각 프로그래밍 시스템은 멀티미디어 검색형(Multimedia Retrieval) 시스템에서 효과적으로 활용될 수 있다.

제2절과 제3절에서는 분산 개발 환경과 객체지향 시각 프로그래밍에 대하여 간단히 설명한다. 제4절에서는 분산 개발 환경에서의 객체지향 시각 프로그래밍이 어떻게 분산되어야 하며, 그 때의 특성과 역할을 설명한다. 제 5 절에서는 분산 객체 관리 환경에서 객체지향 시각 프로그래밍 시스템의 효율적 활용 예로서, 분산 시스템 개발을 지원하는 도구인 VIOLA (VIsual Object-Oriented Language for Multimedia Applications)[1]의 구조와 특성을 설명한다. VIOLA는 RODEO(Research on Object-oriented multimedia application Development Environment on Object management architecture) 시스템의 객체지향 시각 프로그래밍 도구이다. RODEO는 객체 컴퓨팅 표준 구조인 OMA/CORBA와 초고속 통신망

1) VIOLA는 "RODEO : OMA 기반 객체지향 멀티미디어 소프트웨어 개발환경 연구"과제의 소과제인 "객체지향 시각 프로그래밍 도구 개발"로 시스템공학연구소에서 수행되고 있다.

을 기반으로 Video On-Demand 유형의 검색형 멀티미디어 응용 프로그램을 효과적으로 개발할 수 있도록 지원하는 객체지향 소프트웨어 개발 환경이다. 제6절의 결론에서는 앞으로의 연구 방향을 설명한다.

2. 분산 개발 환경

2.1 분산 개발 환경의 특성

객체의 분산 및 확정 가능함과 동시에 소프트웨어 개발의 생산성을 위하여 다음의 특성을 지닌다.

◇ 분리/분산
시스템 내의 객체를 분리/분산하여 객체 접속의 원격성, 일치성 미흡, 분산병행실행, 오류보정, 데이터나 응용프로그램의 이동, 동적 재구성 등을 가진다.

◇ 이형성
이기종 간의 분산 처리 능력을 지니기 위하여는 호환성을 지녀야 하며, 이때 각 연방적인 이름지정, 자료 번역, 객체나 응용프로그램의 보호가 뒤따라야 한다.

◇ 지역적 자치성
분산 객체의 여러 자치적인 권리나 권한의 제어가 필요하며, 각 권한 사이에서의 일치성, 캡슐 상태의 보존 등이 요구된다.

◇ 호환성
객체의 새 버전이나 운영체제, 응용소프트웨어와 데이터의 실행 환경에서의 호완성을 지닌다.

◇ 이식성
다른 시스템의 운영체제에 응용프로그램의 이식이 가능하고 기존 응용프로그램이 활용한다.

◇ 확장성
한 응용프로그램이 여러 규모의 컴퓨터 시스템 사이에서 실행된다.

◇ 상호 운용성
여러 시스템 사이에서 네트워크 상호접속에 의한 통신과 협조가 있다.

2.2 분산 개발 환경의 구조

분산 개발 환경의 구조 모델은 네트워크가 기반을 이루고 있다. 분산 개발 환경의 모델로는 POSIX(Portable Operating System Interface for Computing Environment)의 OSE (Open System Environment), OSF(Open Software Foundation)의 DCE(Distributed Computing Environment), UI(UNIX International)의 ATLAS, OMG(Object Management Group)의 OMA(Object Management Architecture) 등의 여러 모델이 있는데, 분산 개발 환경에서 필요하게되는 사용자의 각 지원 서비스를 통합한 계층 구조의 모델이다. 여기에서 OMG의 OMA는 응용 객체, 공통 기능, 객체 서비스, 객체 요청 중계자 의 각 기능을 가진다. 응용 객체는 OMG 사양의 응용 프로그램이나 클래스이며, 공통 기능은 응용프로그램 간에 객체를 공유하거나 OMA 기반 응용프로그램 생성 요소이다. 객체 서비스는 분산 시스템 내의 객체가 공동으로 사용하는 OMG 정의 인터페이스이며, 객체 요청 중계자는 객체의 요청/응답 전달 메카니즘이다.

2.3 분산 객체 관리 모델

분산 컴퓨팅 원리 중 하나는 지역적으로 처리될 수 없는 객체나 응용프로그램은 다른 지역에 있는 임의의 컴퓨터에서 처리되도록 한다. 분산 객체 관리 모델의 일반적인 개념과 요구사항은 분산 객체 관리 구조(Distributed Objects Management Architecture : DOMA)의 명세를 사용한다[10]. 이 DOMA 분산된 객체 사이의 통신은 객체가 어떠한 상태에 존재하느냐에 따라 그 접속이나 관리기술이 다르다. 객체의 분산 방법은 크게 다음과 같다.

◇ 여러 객체가 한 프로세스에 존재
◇ 여러 객체가 여러 프로세스에 존재
◇ 여러 객체가 여러 시스템에 존재

한 프로세스 내에 여러 객체가 존재하는 경우, 객체지향 응용프로그램은 한 프로세스 내에서 라이브러리로 객체 관리 구조와 연결된다. 여러 프로세스에 여러 객체가 있으면, 객체 관리 구조를 대몬 프로세스와 같이 취급한다. 그러나 여러 시스템 내에 객체들이 분산되어 있으면 진정한 의미의 분산 개발 환경의 구조

를 지원하게 된다. 이때는 분산된 객체를 인식하는 문제 때문에 객체 사이에 IDL를 사용하는 접속 방식을 정의하며, 객체 사이의 요청과 응답을 전달하는 방식이 필요하다.

분산 프로세스들 사이에서 프로세스의 상호 통신은 한 프로세스가 다른 프로세스의 접속을 요청하여 이루어지는데, 이것은 초기화된 프로세스가 원격 프로세스의 위치를 안다는 것을 의미한다. TCP/IP 에서 초기 프로세스는 원격 프로세스가 실행될 컴퓨터의 IP 주소와 네트워크를 기동시키기 위한 포트를 알고 있다. 분산 객체 관리 구조로 DOMA를 사용하면, 처음에는 초기 프로세스가 의미를 가진 문맥을 결정한 후 원격 프로세스의 주소를 제공하기 때문에, 원격 프로세스의 서비스를 인지할 수 있다.

프로세스는 API[3]를 통하여 DOMA와 연결되는데, API는 네트워크의 접속과 완전 분리된다. DOMA의 기본 메카니즘은 네트워크 추상화 계층과 관련이 있는데, 이 계층을 사용하면 서비스의 지원이 투명하게 이루어진다. 그림 1 은 DOMA API의 계층 구조이다.

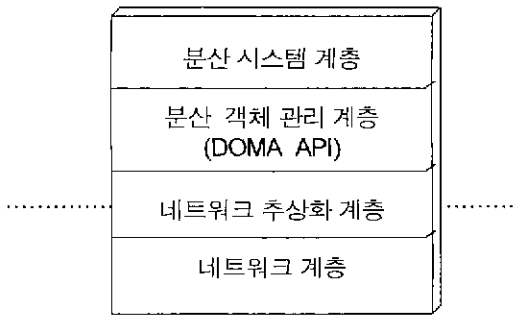


그림 1 DOMA API의 계층 구조

2.4 사용자 인터페이스

분산 개발 환경을 효율적으로 활용하려면 사용자 인터페이스가 중요한 역할을 한다. 특히 시각 프로그래밍 기술을 분산 개발 환경에 적용하려면 사용자 인터페이스의 관리가 분산 객체 관리 모델과 함께 연구되어야 한다. 분산 프로세스가 설계되려면 플랫폼과 사용자 인터페이스로 분리되어 설계되는 것이 효과적이다. 서로 다른 그래픽 사용자 인터페이스(Graphic User Interface : GUI)가 호환되어 실행되어야 하기 때문이다. 즉, Silicon Graphics ELF, 개인용 컴퓨터의 Windows 계열, X-Window 등이 분산 개발 환경에서 같이 실행되어야 한다.

분산 개발 환경에서의 사용자 인터페이스의 호환성을 위하여 사용자 인터페이스 환경 부분과 프로세스 구동 부분으로 분리하여 설계되어야 한다. 그러나 사용자 인터페이스 환경 부분과 프로세스 구동 부분이 분리되더라도, 프로세스 자체의 자료 인터페이스 부분은 개인용 컴퓨터의 Windows 계열, X-Window, Macintosh 등과 같이 다른 기종의 사용자 인터페이스 환경에 따라 재설계되어야 한다. 즉, 프로세스 구동 부분이 이식 가능하여야 하며, 어떠한 사용자 인터페이스 환경이라도 수용할 수 있는 프로세스여야 한다. 이기종 간의 사용자 인터페이스의 호환성은 프로세스의 분산성을 증가시키며 객체지향 시각 프로그래밍의 객체의 사용성을 확대한다.

만약 이기종 사용자 인터페이스가 호완성이 없다면 화면을 리디렉션하는 해결 방법이 필요하다. 분산 프로세스가 특정 사용자 인터페이스 환경을 지닌 새로운 사용자 인터페이스를 동적으로 선택할 수 있도록 하려면, 그림 2 와 같이 물리적 통신 프로세스[5]를 제외한 프로세스를 세부분으로 분리할 필요가 있다.

◇ 프로세스 구동기(Process Driver : PD)
프로세스의 구동을 관리한다.

◇ 인터페이스 관리기(Interface Manager : IM)
사용자 인터페이스를 관리한다.

◇ 인터페이스 뷰어(Interface Viewer : IV)
인터페이스를 화면에 디스플레이하며 사용자와 직접 접속하여 입력 데이터를 처리한다.

프로세스 구동기가 새로운 인터페이스를 실행하게되면, 인터페이스 관리기는 통신 메시지에서 메시지 타입을 추출하며, 인터페이스 뷰어가 화면을 관리하도록 내부의 룩업 테이블을 참조한다. 이 인터페이스 뷰어는 지정된 사용자 인터페이스의 사용자 화면에서 구동된다. 사용자가 화면에 데이터나 어떤 응답이 나오도록 사용자 인터페이스에서 행위를 수행하면, 인터페이스 뷰어는 인터페이스 관리기로 입력

데이터를 전달한다. 이 때 입력된 데이터는 프로세스 구동기가 알 수 있는 포맷으로 변환된다. 프로세스 구동기는 데이터에 따라 적당한 처리를 수행하고, 그 응답으로 무슨 동작을 할 것인가를 결정한다. 즉, 새 인터페이스를 멈추게 하거나 시작하는 일 등을 수행한다. 인터페이스 관리기와 프로세스 구동기 사이의 모든 데이터는 일정한 메시지 포멧을 사용하는데, 무형식이고 동적으로 확장 가능하다.

사용자 인터페이스로부터 프로세스 구동기를 보호하기 위하여 사용자 인터페이스와 독립된 화면 기술 언어(Display Description Language)를 사용할 수 있다. 이것은 추상화 용어의 기술로서 프로세스 구동기를 구성하고, 생성된 새로운 사용자 인터페이스가 인터페이스 관리기와 통신할 수 있게 하는 방법이다. 이러한 방법 중 한 방법이 현재 WWW에서 사용되는 HTML[3]이란 언어인데, HTML은 GUI 에 독립적으로 버튼, 텍스트 박스, 리스트 박스 등과 같은 윈도우 메타포를 사용하여 사용자 인터페이스가 그 내용 결과를 화면에 디스플레이 하도록 한다. 이 메타포는 화면에 표현되어지는 내용들이 융통성과 일치성을 갖도록 많은 컴포넌트를 제공한다.

사용자 인터페이스를 화면에 디스플레이하려면, Netscape나 Mosaic 같은 HTML 뷰어는 GUI 속성에 따라 디스플레이 할 수 있도록 사용자 인터페이스를 기동시킨다. 즉, HTML 뷰어는 GUI에 종속된다는 의미이며, 프로세스 구동기와 인터페이스 관리기는 사용자 인터페이스를 독립시킨다. HTML을 사용하면 사용자 인터페이스를 쉽게 생성할 수 있다.

HTML이 여러 범위의 인터페이스를 표현하고 생성하기에 좋은 언어라 하더라도, 어떤 미디어 포맷이나 디스플레이를 취급하기는 알맞지 않은 경우도 있다. 3D 모델, 각 픽셀, 인터페이스의 각 컴포넌트 사이의 공간적인 관계 등은 기술하기 어렵다. 이런 것을 쉽게 표현하는 GUI 독립 프로토콜인 Virtual Reality Modelling Language(VRML)이 있기도 하며, 인터페이스 관리기에 적당한 뷰어나 택을 추가하여 재모델할 수 있다.

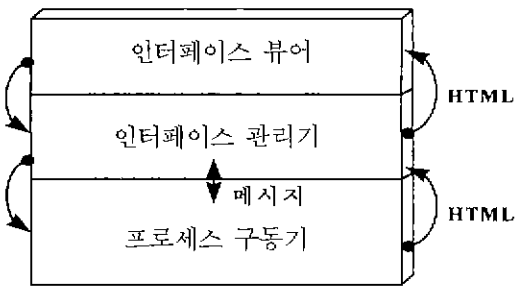


그림 2 사용자 인터페이스

3. 객체지향 시각 프로그래밍

객체지향 시각 프로그래밍은 객체지향 프로그래밍 패러다임을 지원하거나, 객체지향 언어를 시각적인 환경으로 지원하는 경우가 보통이다[15, 16, 17, 18, 19, 22]. 각 경우에서 각 객체에 대한 시스템의 기능과 메시지 전달을 통한 정보 처리 기능을 지원한다. 그 외에도, 시각 객체지향시스템은 클래스나 인스탄스를 사용하는 함수의 재사용 방법을 제공한다. 결국, 객체지향 시각 프로그래밍의 목표는 객체지향 기술의 재사용성, 확장성 등의 특성과 시각 프로그래밍에서의 접근성의 특성을 합쳐 놓은 것이다.

각 특성이나 장점은 이미 여러 연구[15, 16, 17, 18, 19]가 되었으며, 두 경우의 특성을 합치면 더욱 훌륭한 프로그래밍 도구가 될 수 있다. 예를 들어, 객체지향 기술에서 지원되는 함수의 팩토링은 시각 환경을 생성을 더욱 강화하게된다. 또한, 객체의 각 종류는 하나나 여러 객체로 구성되는데, 객체의 상태나 용량을 시각화한다. 그림으로 된 각 표현은 객체가 지원하는 기능을 나타내며, 다이아그래밍 도구는 객체 사이를 연결하여 객체의 관계를 나타낸다. 표면적인 문제뿐만 아니라, 시각 기술도 객체지향 기술과 어떻게 이상적으로 잘 결합할 것인가에 대하여도 많이 연구되고 있다[19, 22].

그러나 객체지향 프로그래밍과 시각 프로그래밍의 여러 특성 중에서 일부는 잘 부합되지 않는 경우도 있다[22]. 잘 부합되지 않는 경우는 주로 객체 크기에 대한 연구이다. 뿐만 아니라 개발하고자 하는 소프트웨어 시스템의 크기가 너무 클 때는 그 한계가 있다. 요즈음에

는 컴포넌트를 기반으로 하는 프로그래밍 기술이 상업적으로 발전되고 있다[22]. 상업적인 경우에는 응용 분야의 최종 사용자가 프로그래밍할 때 기존에 이미 내재되어 있는 여러 객체들을 시각적인 기술로 결합하게 된다.

최종 사용자가 시각적인 여러 아이콘을 나열하는 방식으로 시각 프로그래밍이 가능하도록 한 것은 작은 응용 분야에 적용하기에는 매우 효과적이다[18, 21]. 최종 사용자는 프로그램 실행되는 순서나 방법을 시각적으로 표현함으로써 객체지향 프로그램을 생성한다. 대형 소프트웨어 시스템을 구축하기 위한 큰 시각 프로그래밍 방법은 완벽성, 명확성, 생동성 등이 유지되도록 하여야 하며, 정보, 객체, 처리 등의 시스템 원소 사이의 관계를 대용량에 알맞도록 관리하고 표현할 수 있어야 한다. 또한 추상화, 자료형, 지속성과 같은 기존 프로그래밍 언어의 특징에 새로운 접근 방법이 추가되는데, 새로운 시각 표현법, 시각 프로그램 의미, 효과적인 네비게이션 기술 등이다. 그 외에도 연속적으로 어느 특징을 잘 디스플레이 하는 기술도 필요하다. 이런 기술은 산업 소프트웨어의 시각 프로그래밍 효과를 한층 돋보이게 한다.

객체지향 프로그래밍과 시각 프로그래밍 기술의 결합에서 또 다른 중요 분야는 시각 프로그램에서 널리 사용되는 정확성이나 함축성 같은 것들을 어떻게 객체지향 기술과 결합하는가이다[15, 16, 17, 18]. 객체지향 언어의 강점 중 하나는 클래스나 클래스 우선 순위에 대한 추상화를 지원하는 것이다. 또한, 대부분의 객체들은 상속성, 합성, 파견 등을 통하여 다른 객체와의 관계를 함축적으로 정의한다. 다형성은 객체지향 기술을 시각 기술에 적용할 때 기술적인 문제들에 매우 복잡하게 만든다. 그 이유는 정적으로 프로그램을 나타내게되면 어느 객체가 메시지를 받을 것인지, 또 어느 메소드가 그 메시지에 실행될 것인지를 명확하게 나타내지 못하기 때문이다.

객체의 관리나 사용도 확장하여 분산 환경에서 그 확장성을 더욱 넓힐 수 있어야 한다. 이런 확장 문제는 컴퓨터에 내재된 객체 컴포넌트를 효율적으로 분산, 공유할 수 있도록 분산 개발 환경과 연결하여야 한다.

4. DOMA 기반 객체지향 시각 프로그래밍 기술

본 절에서는 분산 객체 관리 구조(DOMA)를 기반한 객체지향 시각 프로그래밍 기술을 설명한다. DOMA 기반 객체지향 시각 프로그래밍은 단일 사용자 중심의 객체지향 시각 프로그래밍을 어떻게 투명성을 보장하는 분산 프로그래밍으로 확대할 수 있는가 설명한다. 기본적인 아이디어는 객체지향 시각 프로그래밍 기술을 크게 두 부분으로 고려한다.

◇ 사용자 인터페이스에서의 객체지향 시각 프로그래밍 작성 기술

◇ DOMA 와 접속하기 위한 자료 인터페이스의 구성 기술(일반적으로 IDL 정의)

위의 두 경우 모두 시각 프로그래밍 언어로 작성되어야 하며 아이콘 등 시각 프로그램 환경이 구축되어야 한다.

4.1 서비스 요구사항

객체지향 시각 프로그래밍 도구는 기본 컴포넌트에 주어진 기능과 정보 시스템의 링크 서비스 기능을 수행하는 도구이다. 분산 개발 환경에서의 객체지향 시각 프로그래밍 도구를 구축하려면 분산 환경에서의 링크 서비스가 매우 중요한 부분이며 다음과 같은 요구사항을 만족하여야 한다.

◇ 객체의 수와 크기
시스템이 포함할 수 있는 객체의 최대 수나 객체의 크기에 따른 어떠한 제한도 없이 새로운 노드, 링크, 앵커, 다른 객체의 임포트가 가능하여야 한다.

◇ 데이터 포맷
시간 객체를 포함한 어떠한 데이터 포맷도 사용할 수 있어야 한다.

◇ 응용프로그램
분산 개발 환경 내에 있는 어떠한 응용프로그램도 엑세스하여 공유하여야 한다.

◇ 데이터 모델과 구조
객체지향 시각 프로그래밍의 데이터 모델을 구성하고 있는 단일 뷰를 강조하지 않

아야 한다. 다른 시스템 내에 있거나 새로운 데이터 모델이 상호 협동할 수 있도록 재구성하거나 확장 가능하여야 한다. 그러므로 이 데이터 모델은 외부의 객체지향 시각 프로그래밍 시스템과 상호 운용이 가능하여야 하며, 외부의 다른 시스템과 데이터를 교환할 수 있어야 한다.

◇ 플랫폼
분산 플랫폼에서 구현 가능하여야 한다.

◇ 사용자 인터페이스
다중 사용자를 지원하며, 필요할 경우에는 객체를 단일 사용자를 위한 개인 뷰를 유지하여야 한다.

◇ 시각화 수준
객체의 크기와 사용 객체에 따른 시각화 수준을 정의하여야 한다.

◇ 통신 수준
객체의 크기나 응용 소프트웨어의 크기에 따른 통신 메시지의 크기를 정의하여야한다.

기존의 분산 시스템들은 대체로 위의 요구사항을 만족하지만, 객체지향 시각 프로그래밍의 분산 컴퓨팅에서의 서비스 요구사항은 매우 약한 편이다. 그러므로 위의 요구사항을 만족하는 객체지향 프로그래밍 시스템을 계층화하는 효율적인 분산 프레임웍이 요구된다.

4.2 객체지향 시각 프로그래밍 자원의 캡슐화

객체지향 시각 프로그래밍 언어는 아이콘에 의하여 모든 프로그램의 의미와 객체의 관계를 표현하여야 하며, 그 표현된 시각 프로그램의 각 객체 아이콘은 분산 개발 프로그래머의 행위와 이벤트에 따라 자료 인터페이스로 전달된다. 시각 프로그래밍이 가능하도록 아이콘들에 대한 객체나 프로그래밍에 필요한 모든 객체를 관련지어 시각 프로그래밍 자원이라 할 수 있다. 이 자원은 분산으로 정보 공간이 확대되고 필요한 객체와 쉽게 관련지어지도록 캡슐화한다. 객체지향 시각 응용프로그램(Object-Oriented Visual Application)(그림 3)은 서로 관련된 여러 프로세스, 도큐먼트, 링크 데이터, 또 다른 응용프로그램들의 계층화된 객체로서

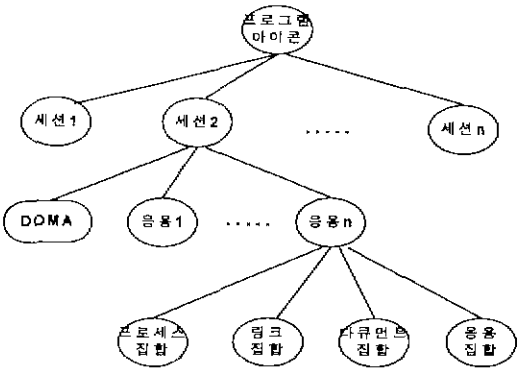


그림 3 객체지향 시각 응용프로그램 자원의 계층구조

분산 환경에서 바인딩될 수 있다고 정의할 수 있다. 이 객체지향 시각 응용프로그램은 지역적으로 혼자 사용되기도 하고 공개될 수도 있다. 그림 3은 일반적인 객체지향 시각 프로그램을 위한 객체 자원을 캡슐화한 계층구조이다. 각 아이콘의 의미에 해당하는 객체는 인터페이스 관리기와 프로세스 구동기에 의해 객체 계층구조와 관련지어진다.

객체지향 시각 응용프로그램의 사용자는 분산 개발 환경 구성의 추상화 레벨에 영향을 받는다. 사용자 인터페이스에서 객체지향 시각 프로그램을 아이콘으로 작성할 때, 여러 응용프로그램을 새롭게 작성할 수 있다. 즉, 응용프로그램 은 가능한 모든 프로세스, 링크데이타, 도큐먼트, 응용프로그램들의 관련에 의하여 관계지어진다. 한 세션은 한 사용자의 프로그래밍 작업 공간에 필요한 시스템 공간으로 여러 객체지향 시각 응용프로그램과 하부구조인 DOMA와의 접속으로 형성된다. 각 사용자는 임의의 객체지향 시각 응용프로그램을 구성하기 위해 한 세션을 구성할 수 있으며 이 세션을 통해 한 분산된 객체지향 시각 응용프로그램을 구성한다. 각 사용자는 여러 세션을 가질 수 있으며, 현재의 세션은 분산되어 실행될 여러 메시지를 생성한다.

4.3 객체지향 시각 프로그래밍 자원의 시각화

객체지향 시각 프로그래밍을 위하여 필요한 자원을 시각화하여야 한다. 시각 프로그램에 언어적 문법 구문과 의미가 주어져야 하는데

이 분야에는 많은 연구가 있다[24]. DOMA의 응용객체, 공통기능, 객체 서비스 등에 대하여 아이콘화하여야 한다. 이 때 DOMA에 접속되기 위한 관련 아이콘은 다음과 같다.

◇ IDL 정의 아이콘
◇ IDL 컴파일 아이콘
◇ 객체 구현 아이콘
◇ 서버 메인 프로그램 아이콘
◇ 객체 구현 준비 아이콘
◇ 서버 등록 아이콘
◇ 서버 기동 아이콘
◇ 객체 바인딩 아이콘

이와 같이 객체지향 시각 프로그램을 분산 환경에서 구동시키기 위하여 분산 객체 관리 구조와 접속하는 과정도 시각 프로그래밍 언어로 정의하여야 한다.

4.4 분산 환경과 객체지향 시각 프로그램의 접속과 바인딩

이미 언급한 바와 같이 사용자 인터페이스에서 분산 응용 프로그램을 시각 프로그래밍 도구로 작성하면 분산 환경과 접속되어 실행한다. 이 때, 분산 개발 환경의 일반적인 객체 관리 구조인 DOMA와의 접속 IDL을 이용하는데 그 과정은 다음과 같다.

(1) IDL 아이콘을 이용하여 응용프로그램에 해당하는 IDL을 정의한다.
(2) 정의된 IDL을 컴파일한다. 컴파일이 실행되면 정의된 IDL에 해당되는 C++ 클래스가 생성되며, 필요한 헤더 파일, 작성된 객체지향 시각 프로그램을 위한 스텁, 서버, 스켈레톤이 생성된다.
(3) 생성된 서버 스켈레톤을 이용하여 객체 구현 프로그램을 생성하며 서버의 메인 프로그램에서 객체 인스턴스를 생성한다.
(4) 구현된 서버 프로그램을 저장하고, 프로세스를 생성하여 대기시킨다. 저장할 때는 대몬 프로세스가 실행한다.
(5) 클라이언트에서 객체지향 시각 프로그램을 작성하여 요청을 기동시킨다. 그 후 클라이언트에서 정적 바인딩 아이콘을 연결한다.

위의 프로그래밍 절차에서 (1)에서 (4)는 분산 환경과 객체지향 시각 프로그램과의 접속을 위한 준비 관계이며 실제 응용프로그램은 (5)에서 작성하고 분산 객체를 요청한다. 클라이언트의 객체지향 시각 프로그램과 서버의 객체를 바인딩할 때는 객체 참조를 위하여 원격지 객체에 대한 프록시가 생성되고, 이 플록시를 이용하여 필요한 오퍼레이션이 기동된다. 물론 오퍼레이션 전에는 반드시 객체의 바인딩이 먼저 이루어진다.

위의 DOMA에 연결되기 위하여 이미 아이콘으로 정의되어 있어 쉽게 접속 과정이 이루어져야 한다. 그림 4는 DOMA를 구성하고 있는 객체의 계층구조와 객체지향 시각 프로그래밍 자원 객체의 계층구조가 접속과 바인딩에 의하여 하나의 새로운 객체를 형성하여 계층구조를 이룬다.

위에서 언급한 과정은 클라이언트의 객체지향 시각 프로그래밍과 분산 환경에서 객체를 바인딩하기 위한 사용자의 프로그래밍 과정이다. 이외에도 실질적으로 객체지향 시각 프로그래밍 모델이 분산 환경에서 동작하기 위해서는 최소한 다음과 같은 컴포넌트가 있어야 한다.

◇ 외부 링크 저장
객체지향 시각 프로그래밍에서 사용자의 요청에 응답하기 위한 링크베이스 프로세스가 모든 요청 링크를 유지하고 질의와 생성을 관리다.

◇ 프레젠테이션과 브라우징
요청과 응답에 대한 사용자 정보의 디스플레이와 사용자의 브라우징은 시스템의 뷰어 클래 취급한다. 뷰어는 사용자와 직접 접속한다.

◇ 객체 선택
사용자가 임의의 객체를 선택하면 객체 연결 상태를 표현하고 난 후, 필요한 정보를 디스플레이하도록 적당한 뷰어를 기동시킨다.

◇ 객체 링크 저작
새로운 객체의 등록 등에 의한 새로운 링크를 생성시킨다. 링크의 시작과 끝의 위치, 링크의 타입, 링크의 설명 등을 말한

다. 이러한 정보가 모아지면 완전한 링크가 생성되도록 링크베이스에 돌어간다.

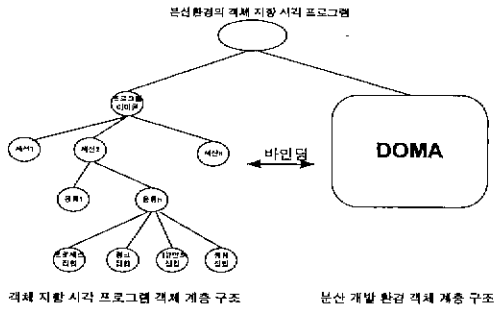


그림 4 시각 프로그래밍 자원과 분산 환경과의 접속

분산 개발 환경에서의 객체지향 시각 프로그래밍 시스템은 이산, 분산된 프로세스들과 각 시각 아이콘의 객체나 원소를 결합하여 그 개념을 모델화한다. 또한 메시지가 이런 프로세스들에 전달, 접수되어 처리된다. 다른 프로세스와 비동기 통신할 수 있으며, 프로세스와 메시지의 분산은 DOMA에 의하여 이루어지는데, 프로세스의 위치와 관계없이 독립적이다.

시스템 내의 자원은 그 자원을 사용하는 응용프로그램 내용에 따라 분산되는데, 일단 사용자가 한 응용프로그램을 사용하게되면 이 과정이 모두에게 공개한다. 분산 개발 환경에서의 객체지향 시각 프로그래밍 시스템은 사용자 세션 내에 포함되어있는 원격지의 응용프로그램이 투명하게 사용하도록 하는 메카니즘을 제공하여야 한다.

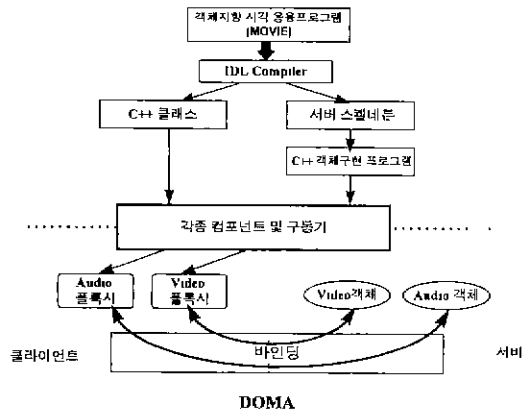


그림 5 객체지향 시각 프로그램 객체의 바인딩

그림 5는 분산 환경과 객체지향 시각 프로그램의 접속과 바인딩에 관한 과정과 요청 메시지의 흐름을 나타낸다. 그림 5는 사용자 인터페이스에서 MOVIE라는 객체지향 시각 프로그램을 구성하기 위한 것인데, 사용자 인터페이스에서 아이콘에 의해 시각적으로 프로그램을 구성하는 과정이다. 이 때, 필요한 객체 Audio와 Video를 분산 환경에서 바인딩한다. 각종 컴포넌트 및 구동기는 각 아이콘의 의미에 따라 구동되거나 실행될 템플리트이다. 그림 5의 각 아이템에 대하여는 이미 설명한 것과 같다.

5. 객체지향 시각 프로그래밍 도구 : VIOLA

본 절에서는 객체지향 시각 프로그래밍의 응용으로 VIOLA에 대하여 설명한다. 이 VIOLA는 분산 개발 환경에서 보다는 분산 객체 환경에서의 "분산 멀티미디어 응용 시스템 개발을 위한 객체지향 시각 프로그래밍 도구" 이다. 즉, 사용자가 자신의 서비스를 위한 시각 프로그램을 구성하고 난 후, 분산 환경에서 그 서비스를 제공받는다. 그러므로 VIOLA는 특정 분야에 국한되지 않고 다양한 종류의 검색형 멀티미디어 응용 시스템을 시각적으로 쉽게 개발할 수 있도록 지원하는 객체지향 시각 프로그래밍 도구이다. VIOLA는 프로그래밍에 대한 깊은 지식이 없는 일반 사용자가 멀티미디어 응용 시스템의 작동 절차만 이해하면 멀티미디어 응용 시스템을 빠르고 용이하게 개발할 수 있다.

향후 VIOLA는 다중 사용자가 소프트웨어 개발에 필요한 부품들을 공유할 수 있는 팀 프로그래밍을 지원할 계획이다. 네트워크 기반의 개발 환경에서 클래스 저장소에의 동시 접근을 지원함으로써 비동기적으로 공동의 소프트웨어 개발이 가능하도록 할 것이다.

5.1 VIOLA 모형

VIOLA는 정보통신 국책 연구 개발 사업으로 한국전자통신연구소와 시스템공학연구소가

공동으로 수행중인 RODEO(Research on Object-oriented multimedia application Development Environment on Object management architecture) 시스템의 객체지향 시각 프로그래밍 도구이다. RODEO는 객체 컴퓨팅 표준 구조인 OMA/CORBA와 초고속 통신망을 기반으로 Video On-Demand 유형의 검색형 멀티미디어 응용 프로그램을 효과적으로 개발할 수 있도록 지원하는 객체지향 소프트웨어 개발 환경이다.

RODEO는 분산 멀티미디어 공통 기(DMCF : Distributed Multimedia Common Facility), 멀티미디어 객체 부품 저장소 및 재사용 시스템(RESMO : REuse System for Multimedia application Object), 객체 지향 시각 프로그래밍 도구(VIOLA)로 이루어져 있다. DMCF는 분산 객체 컴퓨팅 환경인 CORBA를 기반으로 이기종 간의 검색형 멀티미디어 서비스를 가능하게 하고, 분산 데이타베이스를 통합 관리하는 브로커를 제공함으로써 사용자에게 투명성을 제공한다. RESMO는 멀티미디어 객체 모델링 도구, 부품 품질 평가 도구, 재사용 부품 저장소, 저장소 관리 도구, 부품 검색, 수정 합성 도구 등으로 구성된다.

VIOLA는 클라이언트와 서버의 DMCF를 이용하여, 데이터베이스나 서버에서 제공하는 서비스를 이용하게 된다. VIOLA는 서버와의 통신을 위하여 TCP/IP를 사용하거나 스트림의 다운로드를 위해 ATM 버스를 사용한다. 클라이언트 시스템은 웹 환경과 자바스크립트를 지원함으로써 하드웨어와 운영체제에 독립적으로 구성될 수 있다. 즉, 대상 시스템은 자바지원 웹 브라우저와 DMCF의 클라이언트 모듈만 있으면 어떠한 플랫폼에서도 작동 가능하다.

5.2 패러다임

VIOLA는 객체지향 프로그래밍, 컴포넌트 기반 프로그래밍, 데이터 흐름 방식의 아이콘 기반 시각 프로그래밍 패러다임을 사용한다.

(1) 객체지향 프로그래밍

멀티미디어 프로그래밍에서는 은닉성, 모듈성, 확장성이 요구되므로 이를 위해서 VIOLA는 객체지향 프로그래밍을 지원한다. 멀티미디어 응용 시스템 개발자에게 특정 미디어와 하드웨어의 세부적인 사항을 은닉하고, 미디어 처리의 복잡한 인터페이스를 모듈화된 형태로 생성한다. 기존의 코드를 개선하고 확장하는 방법을 제공함으로써 멀티미디어 응용 시스템의 요구사항 변화에 적응할 수 있다.

(2) 컴포넌트 기반 프로그래밍

객체지향 프로그래밍과 시각 프로그래밍은 개발 시스템의 규모 면에 있어서 차이를 보인다. 객체지향 프로그래밍은 대규모의 시스템 개발에 적합하고, 시각 프로그래밍은 소규모의 시스템 개발에 적합하다. 이러한 차이를 극복하기 위해 컴포넌트 기반 프로그래밍은 객체지향 개념을 포함한 컴포넌트를 이용하여 시각 프로그래밍을 가능하게 함으로써 사용자 관점에서는 대규모 시스템을 마치 소규모의 시스템을 용이하게 개발하는 것처럼 보이게 한다.

멀티미디어 요소와 처리 루틴들을 객체지향 개념의 한 컴포넌트로 정의하고, 그 컴포넌트들을 연결함으로써 멀티미디어 작동 절차를 표현한다. 검색형 멀티미디어 응용 시스템 개발에 필요한 컴포넌트에는 사용자 접속, 멀티미디어 검색, 멀티미디어 프리젠테이션에 관한 기능들을 포함하여야 한다.

(3) 자료흐름 방식의 시각 프로그래밍

멀티미디어의 작동은 멀티미디어의 자료 흐름을 통해 쉽게 표현될 수 있다. 이를 시각적으로 프로그래밍하도록 함으로써 멀티미디어 프로그래밍에 대한 깊은 지식이 없는 사용자도 멀티미디어 데이타 검색과 프리젠테이션에 대한 프로그래밍을 할 수 있다.

5.3 인터프리터 구성

VIOLA는 그림 6과 같이 시각 프로그래밍 편집기, 시각 언어 번역기, 시각 프로그램 실행기, 정보저장 관리기, DMCF API 접속기로 구성된다.

시각 프로그래밍 편집기는 그래픽 사용자 접속, 데이타베이스 접속, 멀티미디어 접속 제어

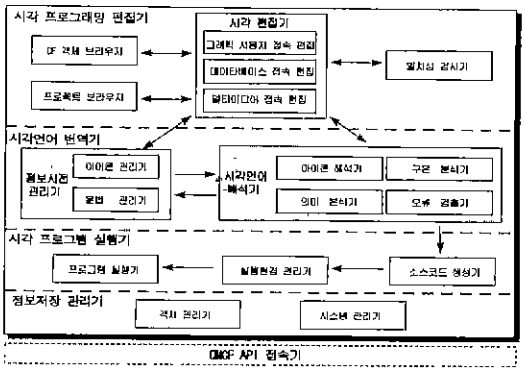


그림 6 VIOLA의 구성

의 시각적 편집, 시각 객체간의 연결에 대한 일치성 검사, DMCF API 라이브러리 객체에 대한 브라우징, 프로그램의 전체 구성에 대한 검색을 지원한다.

시각 언어 번역기는 시각 객체, 시각 언어 및 문법에 대한 정의와 관리를 하고, 시각 문법에 따라 시각 프로그램을 인식, 분석 및 해석한다.

시각 프로그램 실행기는 시각 프로그램의 번역 결과로부터 HTML과 Java 코드를 생성하고 서버로의 uploading과 실행을 위한 환경변수를 설정하여 검색형 멀티미디어 응용 시스템을 실행시킨다.

정보저장 관리기는 시각 객체의 정보를 저장 및 관리하고 VIOLA에 대한 사용자 등록과 접근 제한 등 시스템을 관리한다.

DMCF API 접속기는 VIOLA와 DMCF API 라이브러리 객체와의 접속을 지원한다.

5.4 VIOLA의 기능

VIOLA는 분산 환경하의 멀티미디어 데이터를 검색하여 클라이언트 시스템에 출력하고 제어하는 프로그램을 시각적으로 개발하기 위하여 다음과 같은 기능을 가진다.

(1) 다양한 그래픽 사용자 접속

서버와의 통신, 특정 미디어의 검색, 선택된 미디어의 프리젠테이션 및 제어에 필요한 GUI 개발 기능을 제공한다.

(2) 서버와의 통신

서버와의 접속 및 해제, 제어 신호의 전송 및 결과의 수신을 시각적으로 프로그래밍하는 기능을 제공한다.

(3) 효율적인 멀티미디어 정보 검색

다양한 멀티미디어 정보 검색 형태의 프로그래밍 기능, 검색된 자료의 브라우징을 위한 프로그래밍, 동적 자료 선택을 위한 프로그래밍 기능을 포함한다.

(4) 미디어 재생과 제어

선택된 결과의 정적 미디어를 사용자가 원하는 형태로 출력하는 기능, 오디오/비디오 스트림과 같은 동적 미디어에 대한 제어, 미디어간의 동기화 명세 기능(예 : 비디오 스트림에 광고를 삽입하는 경우, 여러 개의 비디오를 순차적으로 상영하는 경우)이 포함된다.

(5) 예외 사항 처리

서버, 통신망, 또는 클라이언트 시스템의 장애나, 사용자의 조작 실수, 데이터 포맷의 불일치 등으로 일어날 수 있는 에러를 처리하는 기능을 프로그래밍을 할 수 있다.

VIOLA의 대상 응용 분야는 DMCF API로 개발할 수 있는 검색형 멀티미디어 응용 시스템의 클라이언트 프로그램으로서, Movie On-Demand, Video Kiosk, 홈쇼핑, 홈뱅킹, News On-Demand, Game On-Demand, Karaoke On-Demand, 가상 CD-ROM, TV Listing 등이 있다.

6. 결 론

지금까지의 시각 프로그래밍 기술과 객체지향 프로그래밍 기술은 각기 따로 발전되어왔으나 두 기술이 결합되어 효율적인 객체지향 시각 프로그래밍 기술로 발전하였다. 뿐만 아니라, 분산 환경의 발전으로 공동 프로그래밍 및 분산 프로그래밍 기술의 접목으로 좀더 넓은 정보 공간을 가지게되었다.

분산 환경 시스템에서 진정한 의미의 분산 시스템을 개발할 수 있는 분산 객체 관리 구조

와 모델에 의하여 효율적인 객체지향 시각 프로그래밍의 구성할 수 있다. 이렇게 하기 위하여 진정한 분산성을 제공하는 분산 객체 관리 모델이 제안되어야 하며, 이 분산 개발 환경을 효과적으로 활용할 수 있는 시각 프로그래밍 기술에 대하여 설명하였다.

단순 시각 프로그래밍과 달리 객체의 여러 특성과 시각적인 의미를 관련지어야하며, 그러한 객체지향 시각 프로그래밍이라 하더라도 분산 환경과 접속하는 과정도 쉬운 일은 아니기 때문에 그 접속과정의 시각 프로그래밍화를 개략적으로 간단하게 제시하였다.

이 글에서 서술된 DOMA 기반 객체지향 시각 프로그래밍 시스템은 분산 시스템을 구성하기 쉽게 하고, 다른 분산 객체지향 시각 프로그래밍 시스템에서 제공하지 않는 융통성을 제공한다. 즉, 사용자가 객체의 관련성을 모르더라도 시각적인 아이콘에 의해 그 투명성을 제공받는다. 또한 자원의 시각화에 의해 사용자 사이에서의 공동작업은 정보의 공유를 강하게 지원하며, 특별히 각자의 자원을 분산 프로세스의 그룹으로 캡슐화할 수도 있다. 객체지향 시각 응용프로그램은 자세한 것은 물론 새로운 응용프로그램을 만들거나 구성하기 쉬운 메카니즘을 제공한다.

앞으로의 연구 방향은 Java[23]의 특성을 살려 좀더 융통성, 이기종성, 이식성을 지닌 객체지향 시각 프로그래밍 프로세스를 개발한다. 이 프로세스는 다중 플랫폼을 지원하는 동적 데이터 타입을 제공하도록 Java의 객체지향 특성을 가진다. 또한 DOMA는 사용자의 각 세션에서 이벤트를 브로드케스팅하고 공동작업을 수행할 수 있도록 설계되었기 때문에, 좀더 나은 CSCW[19]를 지원하는 하부 구조가 될 것이다. 원격지의 사용자에게 자원을 공개할 때는 보안 문제를 고려하지 않을 수 없다. 그러므로 사용자의 권한 제한 메카니즘이 절대적으로 필요하며 앞으로의 연구에 추가된다.

DOMA와 VIOLA가 인터넷과 에이전트 기술에서 훌륭한 프레임을 제공한다. Common Object Request Broker Architecture(CORBA)[9]나 Distributed Computing Environment (DCE)[10]와 같은 기술은 분산 응용프로그램의 상호 운용을 보여주는 좋은 기술이므로, 이 VIOLA 시스템도 현재의 표준화된 기술이나 앞으로의 연구 기술에 통합되어 한 부분에 적용될 것이다.

참고문헌

[1] Andrews, K., Kappe, F. and Maurer, H., Serving Information to the Web with Hyper-G, Computer Networks and ISDN Systems, Vol. 27, No. 6, pp. 919-926, 1995.

[2] Berners-Lee, T., Cailiau, R., Groff, J. and Pollermann, B., World-Wide Web : The Information Universe. In : Electronic Networking : Research, Applications and Policy, Vol. 2 No 1, Spring, Meckler Publishing, Westport, CT, USA, pp. 52-58, 1992.

[3] Ian S. Graham, HTML Sourcebook, 2nd Edition, John Wiley & Sons, Inc., 1996.

[4] Burger, A. M., Meyer, B. D., Jung, C. P. and Long, K. B., The Virtual Notebook System. In : Hypertext 91, Proceedings of Third ACM Conference on Hypertext, San Antonio, Texas, USA, ACM Press, pp. 395-402, 1991. 12.

[5] Dale, J. D., Distributed User Interfaces : Achieving User Interface Heterogeneity in a Distributed Environment, Multimedia Technical Report M96/1, Department of Electronics and Computer Science, University of Southampton, 1996.

[6] Open Software Foundation, OSF/Motif Programmer's Guide : Revision 2.0, Cambridge MA 02142, 1994. 8.

[7] Hill, G., Wilkins, R. J. and Hall, W., Open and Reconfigurable Hypermedia Systems : A Filter Base Model, Hypermedia, Vol. 5, No. 2, pp. 103-118, 1993.

[8] Nye, A., Xlib Programming Manual Volume 1 (3rd edition), O'Reilly & Associates Inc., 1993.

[9] Object Management Group, The Common Object Request Broker : Architecture and Specification, OMG Technical Document, No. 91-12-1, Revision 1.1, 1991, 12.

[10] 성재모, "DCE를 이용한 분산 컴퓨팅", 한국정보과학회지, 제 14 권, 제 1 호, pp. 24-31, 1996. 1.

[11] PearlL, A., Sun's Link Service : A Protocol for Open Linking. In : Hypertext '89 Proceedings, pp. 137-146, 1989. 11.

[12] Shipman, F. M. Iii, Chaney, R. J. and Gorry, G. A., Distributed Hypertext for Collaborative Research : The Virtual Notebook System. In : Hypertext '89 Proceedings, pp. 129-136, 1989. 11.

[13] Smith, K. E. and Zdonik, S. B., Intermedia : A Case Study of the Differences Between Relational and Object-Oriented Database Systems, OOPSLA' 87 Proceedings, pp. 452-465, 1987. 11.

[14] Will, U. K. and Leggett, J., Hyperform : Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. In : D. Lucarella, J. Nanard, M. Nanard and P. Paolini, eds ECHT "92, Proceedings of the Fourth ACM Conference on Hypertext, ACM Press, pp. 251-261, 1992. 11.

[15] Ephraim P.Glinert, Steven L.Tanimoto, 'Pict : An Interactive Graphical Programming Environment," IEEE Computer, Nov. 1984.

[16] Paul Harmon and brian Sawyer,, ObjectCraft : A Graphical Programming Tool for Object-Oriented Applications, Addison-Wesley Publicshing Company, 1990.

[17] N. C. Shu, 'Visual programming : Perspective and approaches," IBM Systems Jounal, Vol. 28, No. 4, 1989.

[18] Shi-Kuo Chang, Principles of Visual Programming Systems, Prentice-hall, inc, 1990.

[19] 김상욱, 구경민, 김만수, 박지은, 서정민, 서호연, 이춘희, "객체지향 프로그래밍을 위한 시각 화 시스템", 한국정보과학회논문지, 제 20권, 제 12호, pp. 1773-1792, 1993. 12.

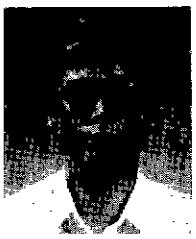
[20] Oxborrow E.A., 'An Object-Oriented approach to Distributed database management systems," DATABASE TECHNOLOGY Vol. 3, No.1, 1990.

[21] T.CJones, 'Reusability in Programming : A Survey of the state of the At," IEEE Trans. on Software Engineering, Vol. SE-10, No. 5, pp. 488-494, Sep. 1984.

[22] 김상욱, 신영길, 이정태, "공유 객체를 지원하는 객체 중심 시각 프로그래밍 환경의 개발", 한국정보과학회논문지(C), 제 2권, 제 2호, pp. 130-141, 1996. 6.

[23] Paul M. Tyma, Gabriel Torok and Troy Downing, JAVA : Primer Plus, Waite Group Press, 1996.

[24] 김상욱, 진윤숙, "공동 시각 프로그래밍 언어", 프로그래밍 언어 연구회지, 한국정보과학회, 제 9 권, 제 2 호, pp. 24-31, 1995. 10.



김 상 욱

1979 경북대학교 전산과학 학사 학위 취득
1981 서울대학교 전산과학 석사 학위 취득
1989 서울대학교 전산과학 박사 학위 취득
1988～현재 현재 경북대학교 컴퓨터과학과 교수로 재직중
관심분야 : 컴퓨터 언어, 객체중심 컴퓨팅, 시각언어, 멀티미디어와 지식처리임.

진 윤 숙

1993 경북대학교 전자계산학과 졸업(학사)
1996 경북대학교 컴퓨터과학과 졸업(석사)
1995～현재 시스템공학연구소 소프트웨어연구부 연구원
관심분야 : 시각 프로그래밍, 객체지향 컴퓨팅, 소프트웨어 엔지니어링