재사용이 용이한 객체지향 기반 협력 시스템 설계 및 구현

허성호[O] 이승룡
경희대학교 컴퓨터공학과
{honet, sylee}@oslab.kyunghee.ac.kr

Design and Implementation of Object-Oriented Based Collaborative System for Reuse

Seong-Ho Heo[0] Sungyoung Lee
Dept. of Computer Engineering, KyungHee University

요 약

협력 시스템은 컴퓨터의 성능 향상과 네트워크 기술의 발전으로 인하여 분산 환경에서 다수의 사람들이 프로젝트나 어떤 작업을 동시에 수행이 가능하도록 하는 기술이다. 현재 대부분의 협력 시스템은 특정 협력 작업에 맞게 개발되어져 왔기 때문에 통합 환경을 제공하기 어렵고, 새로운 협력 작업에 따라 시스템을 확장하는데 많은 시간과 비용이 들어가는 등의 어려움이 있다. 본 논문에서는 객체지향 방법론을 사용한 모듈별 컴포넌트화에 따른 재사용성이 용이한 협력 시스템을 제안하여 일반화, 상세화의 관계나 상속 구조를 통해 클래스의 구현 사항을 재사용 할 수 있도록 하며, 재사용 가능한 모듈을 패키지 형태로 묶어 라이브러리화하여 재사용과 유지보수가 용이하도록 하였다.

1. 서 론

최근 네트워크의 발달과 컴퓨터 환경의 개선으로 컴퓨터 지원 공동작업(CSCW : Computer Supported Cooperative Working)인 원격 화상회의, 공동저작, 공동설계, 전자결제 등의 협력 시스템 요구가 증대되고 있다. 지금까지 개발되어진 협력 시스템은 크게 두 가지 종류로 분류할 수 있는데, 시간적인 접근에 따라 동시에 시스템에 접근할 경우에는 동기적 시스템(synchronous cooperation)이라 하여 화상회의, 네트워크 게임 등이 이에 속하며, 서로 다른 시간에 시스템에 접근할 경우에는 비동기적 시스템(asynchronous cooperation)이라 하여, 메일 시스템, 전자결제 시스템 등이 이에 속한다. 그런데 이들 시스템은 성격이 서로 다른 환경에서 개발되어 왔기 때문에 이들 간의 통합 환경을 제공하기 어렵고, 시스템의 유지보수에 많은 어려움이 있으며, 협력 작업에 따라 새롭게 등장할 협력 시스템에 대한 확장에 많은 시간과 비용이 들어간다. 이러한 문제를 해결하기 위해 소프트웨어의 재사용성에 대한 많은 연구가 진행되어 왔다. 소프트웨어의 재사용은 시간과 비용을 줄여서 생산성과 품질을 향상시키고 다른 소프트웨어를 개발할 때의 지식을 공유할 수 있어 신뢰도를 높여주는 등의 많은 장점을 가지고 있다[1][2]. 이러한 소프트웨어의 재사용은 소프트웨어의 생산성과 신뢰성을 증진하는 우수한 방법으로 등장하면서 소프트웨어 공학의 중요한 관심사로 대두되어 있다.

본 논문에서는 협력 시스템의 확장을 고려하여 객체지향 방법론을 사용한 모듈별 컴포넌트화에 따른 재사용성이 용이한 협력 시스템을 제안한다. 설계시 재사용성을 위해 일반화, 상세화의 관계나 상속 구조를 통해 클래스의 구현 사항을 재사용할 수 있도록 하며, 재사용 가능한 모듈을 패키지 형태로 묶어 라이브러리화하여 재사용과 유지보수가 용이하도록 하였다. 본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고, 3장에서는 협력 시스템을 설계를 설명한 후, 4장에서는 실제 구현된 시스템을 기술, 5장에서 결론 및 향후과제를 서술한다.

2. 관련연구

2.1 협력 작업의 분류

협력 시스템은 시스템이 지원하는 상호 작용과 사용자의 지리적인 분산 정도에 따라 분류된다. 상호작용이 발생되는 시각에 따라, 다른 시각에서 발생되는 비동기화(Asynchronous)와 같은 시각에 발생되는 동기화(Synchronous)로 나누어지고, 지리적인 분산정도에 따라, 같은 장소에 함께 배치되는 대면적(Co-Located)과, 다른 방이나 빌딩 등과 같이 다른 장소에 배치되는 원격적(Remote)이 있다[3]. 대면적 동기화로는 대면회의, 프리젠테이션 등이 있고, 원격적 동기화로는 원격회의, 대면/원격 비동기화는 전자메일, 스케줄관리, 비동기 토론 등이 있다.

2.2 소프트웨어의 재사용성

소프트웨어 재사용은 새로운 소프트웨어 생산과 유지보수에 있어서 기존의 소프트웨어 구축 가공물로부터 표준화된 공통 부분과 잘 알려진 프로세스의 사용을 극대화함으로써 품질과 생산성을 향상시키며 유지보수에 능동적으로 대처할 수 있도록 하는 계획되고 체계화된 행위들의 집합이다. 소프트웨어 개발 과정에서, 특히 객체지향 개발 방법론에 기초하여 재사용의 체계적인 행위는 두 가지 범주로 나눌 수 있다[4]. 먼저, 재사용을 위한 개발(Develop for reuse)은 고수준의 재사용 가능성을

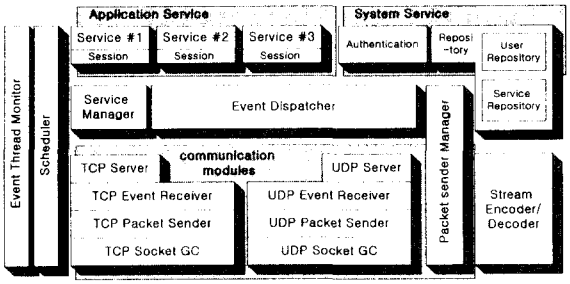
갖는 소프트웨어를 생산하는 것으로 프로젝트 과정 중에 내부에 고수준의 재사용 잠재성을 내재하고 표현할 수 있도록, 개발될 소프트웨어의 초기 식별을 가능케 하는 행위이다. 그리고 재사용에 의한 개발(Develop by reuse)은 이미 존재하는 소프트웨어 컴포넌트와 가공물의 재사용을 극대화하는 것으로 다시 'Reuse as-is'와 'Reuse with change'로 구분될 수 있다. 전자는 변경 없이 그대로 재사용하는 것이며 후자는 수정(modification), 일반화(generation), 조합(composition) 과정을 통해 이루어진다.

3. 협력 시스템 설계

본 논문에서의 협력 시스템 구조는 클라이언트/서버 환경에서 중앙 집중형 방식의 구조를 기본으로 하였지만, 추가될 협력 작업의 서비스 어플리케이션의 규모와 중앙서버의 부하에 따라서 서버를 확장, 분산형 방식으로 할 수 있도록 하였다. 또한 재사용이 용이하도록 하기 위해 객체지향 방법론을 사용한 모듈별 컴포넌트 구조로 설계를 하여, 협력 작업의 확장시 추가해야 할 Service 모듈을 재사용을 통해 쉽게 개발할 수 있도록 하였다.

3.1 서버

서버 시스템은 협력 작업에 따라서 재사용을 통한 시스템의 확장이 용이하도록 하기 위해 [그림 1]과 같이 각각의 모듈로 구성하였다. 이러한 모듈의 구성은 나중에 추가될 협력 작업에 따라서 필요한 Service를 Service할 모듈만 개발하여 Application Service 단에 추가만 하면 되는 장점이 있다.



[그림 1] 협력 시스템 서버의 모듈 구조

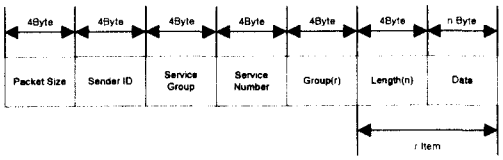
통신 모듈(communication modules)은 클라이언트와 서버간 통신을 지원하며, 클라이언트의 접속을 유지 관리한다. 통신은 소켓을 통한 TCP 방법과 UDP 방법을 사용한다. 먼저 각각의 모듈의 기능을 살펴보면 TCP/UDP Server는 각각의 Connect를 관리하고, TCP/UDP Event Receiver는 접속된 사용자의 Event 입력을 감지하며, TCP/UDP Packet Sender는 접속된 사용자에게 Event를 전송한다. 또한 TCP/UDP Socket GC는 접속된 사용자 중 일정시간 동안 Event를 발생하지 않았을 경우 접속을 종료 시키며, Packet Sender Manager는 보낼 패킷을 관리, 전송 방법에 따라 TCP Packet Sender 또는 UDP Packet Sender에게 보내어 주는 역할을 한다.

Event Dispatcher는 모든 Service를 등록해 놓은 Service Manager 참조하여 TCP/UDP Event Receiver로부터 수신된 Event들을 Service Group에 따라서 분류하여 Service 모듈로 보내는 역할을 한다. 또한 Service Manager는 Service의 등록 및 해제와 현재 서버가 서비스 할 수 있는 Service 모듈이 어떤 것인지 알려주는 역할을 하며, Scheduler는 모든 Event들을 동등하게 처리할 수 있도록 Round Robin 방식의 스케쥴링을 하는 역할을 한다.

Repository 모듈은 정보의 저장소 역할을 하는 부분으로 User Repository와 Service Repository로 구성되어 있다. User Repository는 등록된 사용자의 ID, 패스워드, Key, IP, 접속방식에 따를 레퍼런스 객체(TCP/UDP)등을 관리하고, 접속된 클라이언트의 정보를 저장 및 관리할 수 있도록 설계되었으며, Service Repository는 등록된 Service의 정보를 저장하는 등의 역할을 한다. 클라이언트가 접속했을 때, 클라이언트에 대한 ID, 접속시간, IP등으로 초기화하고, 접속자가 서비스를 이용할 때마다 현재 어떠한 Serivce 모듈을 이용하고 있는지 기록하여 접속자의 상태를 지속적으로 관리할 수 있게 하였다.

Event Thread Monitor 모듈은 서버의 Thread 상태를 모니터링하기 위하여 Java의 Swing을 사용하여 개발한 사용자 인터페이스이며, 서비스 모듈(Service #)은 채팅, 화이트보드, 쪽지 등의 응용 프로그램 부분이며, 클라이언트로부터 전달되어 온 Event를 처리해 주는 Event 처리 모듈이다.

서버간과 서버와 클라이언트가 통신을 위해 [그림 2]와 같이 패킷을 정의하였으며, [표 1]은 패킷의 각 부분의 설명이다. 패킷의 Service Group과 Service Number를 정의하여, 서비스를 구분하는 2가지의 명령어를 가진다. Service Group과 Service Number를 사용하여 서버와 서버 사이의 통신과 서버와 클라이언트간의 통신 프로토콜을 정의한다.

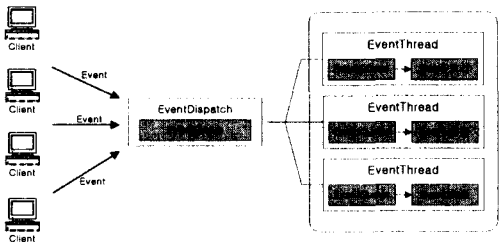


[그림 2] 패킷 구조

[표 1] 패킷 설명

	설 명
Packet Size	패킷 전체 크기
Sender ID	보낸 사람의 IP Address
Service Group	패킷이 처리되어야 할 모듈에 대한 정보
Service Number	패킷 처리시 사용되는 옵션
Group	Item의 개수를 나타낸다.
Length	Data의 길이 정보
Data	실제 데이터

[그림 3]은 Event 처리 모델로 클라이언트에서 전달된 Event는 EventDispatch내의 EventQueue인 FIFO Queue에 삽입되었다가, Event가 처리되는 EventThread에게 해당 Event를 전달하여 처리하도록 하였다. EventThread는 MainScheduler에 의하여 Event를 처리하게 되는데, MainScheduler는 모든 Event들을 동등하게 처리 할 수 있도록 Round Robin 방식의 스케쥴링 방식으로 개발하였다. 여러 EventThread가 동시에 다양한 작업들이 가능하도록 하고, EventThread가 Event를 다른 EventThread로 전파시킬 수 있도록 EventDispatch 기능을 제공하여, 서로 충돌이 발생되지 않는 Event 병행 처리 방법을 제공한다.

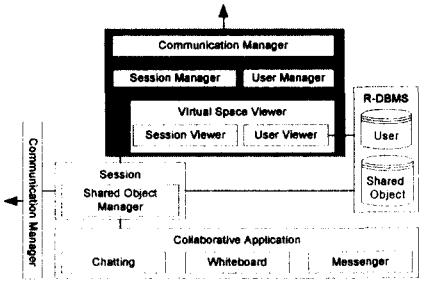


[그림 3] Event 처리 모델

이렇게 모듈별로 설계된 협력 시스템 서버는 추가될 Service 모듈을 재사용을 통해 쉽게 개발할 수 있고, 유지보수에 용이하다는 장점을 가지고 있다.

3.2 클라이언트

클라이언트 시스템은 서버로부터 디자인 프로세스에 의한 스케줄 정보를 획득하여, 현재 협력 작업의 진행상황을 인지하고, 다수의 참여자와 디자인 협력 작업을 지원한다. 이를 위해 클라이언트 시스템은 [그림 4]에서 바탕이 회색인 협력 작업 준비 단계 모듈과 협력 작업 단계로 나누어져 있다.



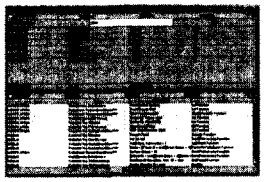
[그림 4] 협력 시스템 클라이언트

협력 작업 준비 단계에서는 서버에 접속하여 Repository를 통해 타 사용자들의 접속유무와 협력 작업 진행 상황들을 사용자에게 제공한다. 이는 Communication Manager를 통해 입력된 패킷의 이벤트 객체로 만들어 Session Manager와 User Manager에서 처리하여 사용자 인터페이스를 통해 제공된다. User Manager는 타 사용자의 접속 유무에 대해 정보를 제공하며, 사용자 검색, 사용자에게 메시지 보내기 등의 서비스를 제공한다. Session Manager는 현재 협력 작업 상황을 제공하며 세션 생성, 참여 서비스 요청 방법을 제공한다. Virtual Space Viewer는 사용자 정보 및 세션에 대한 정보를 출력하는 사용자 인터페이스 모듈이다.

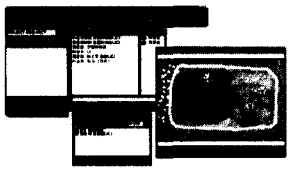
협력 작업 단계는 세션 생성 또는 참가를 통해 타 사용자와 데이터를 공유하여 작업을 수행함을 의미한다. Session의 참여는 서버로부터 접속 인증을 받은 후, Session의 접속에 필요한 정보를 수신하여, 서버의 Session에 참여한다. Session 모듈은 세션의 참여시 필요한 정보를 서버의 Service Repository로부터 전송 받아 세션에 접속 기능을 수행하고, Collaborative Application 모듈은 협력 작업 시 사용되는 응용 프로그램에 대한 인터페이스를 제공한다. 현재 구현된 응용 프로그램은 채팅, 화이트보드, 쪽지 등이 있다.

4. 구현

[그림 5]는 Java의 Swing을 사용, 서버 상태를 모니터링하기 위해 구현한 것으로, Thread Monitor에서는 현재 서버 내에서 작동하고 있는 Event Thread들의 목록을 보여주고 있으며, 그 Thread들의 현재 상태를 실시간으로 모니터링 할 수 있도록 하였다. Event Log는 서버에서 어떠한 작업이 진행되어 질 때, 어떠한 모듈에서 작동되고 있으며, 전달되는 데이터는 무엇인지 실시간으로 모니터링 할 수 있도록 개발하였다. 또한 Scheduler에서 라운드 로빈 방식의 스케줄링을 하기 위해 필요한 Time Slice 값을 설정할 수 있도록 하였다. [그림 6]은 현재 개발한 채팅, 화이트보드, 쪽지의 Service를 제공하고 있는 클라이언트의 그림이다. 클라이언트는 처음에 로그인의 사용자 인증을 거친 후에 서버에 접속하며, 사용자가 필요로 하는 서비스들을 서버에 요청하여, 그에 따른 서버의 응답에 맞추어 작동되고 있다.



[그림 5] 서버 구현 화면



[그림 6] 클라이언트 구현 화면

5. 결론 및 향후 연구

본 논문에서는 확장할 협력 작업들을 쉽고 편리하게 개발하여 추가 삭제가 용이하도록 하기 위해 객체지향 방법론을 사용한 모듈별 컴포넌트화에 따른 재사용성이 용이한 협력 시스템을 설계하고 구현하였다. 서버는 Java, 클라이언트는 Visual C++로 구현하였고 재사용성을 위해 일반화, 상세화의 관계나 상속 구조를 통해 클래스의 구현 사항을 재사용 할 수 있게 구현하였으며, 또한 재사용 가능한 모듈을 라이브러리화하여서 필요에 따라 사용할 수 있도록 하였다. 이렇게 구현된 협력 시스템은 개발의 시간 단축과 유지, 보수 등에서 상당한 비용 절감을 가져올 수 있는 장점을 가지고 있다. 실제 이 협력 시스템에서 협력 작업 서비스를 추가하는데 많은 시간과 비용을 절약할 수 있었다.

향후 연구로는 다양한 협력 시스템을 지원하기 위하여 통신 프로토콜 및 서비스 모듈의 계속적인 개발이 필요로 되어지며, 많은 웹 사용자들을 고려하여 웹과의 연동을 지원할 수 있도록 이후의 개발이 필요하다.

<참고문헌>

[1] Liesbeth Dusink and Patrick Hall(Eds), Software Re-use, pp. 1, Springer-Verlag, 1989.
[2] T. C. Jones, Reusability in Programming: A Suvey of the State of the Art, IEEE Trans. on Software Engineering, Vol. SE-10, No. 5, pp. 488~494, Sep. 1984
[3] Tom Rodden, A Survey of CSCW System, Interacting with computer, 1991, Vol. 3, No. 3, pp. 319~352, 1998
[4] E. S. Garnett and J. A. Mariani, "Software Reclamation," Software Engineering journal, pp. 185~191, May. 1990.
[5] 궁상환, 황승구, Collaborative Computing의 기술 및 응용, 정보과학회지 제16권 제7호, 1998. 7.