CORBA based Dynamic Collaborative Work System
(CORBA 기반의 동적 상호협력 공동 작업 시스템)

이세훈*
(Seihoon Lee)

ABSTRACT

Computer Supported Cooperative Work systems for cooperation in distributed computing environment is used in many fields such as tele-conferencing, co-authoring, on-line approval, and so on. But, because of designing of no considering about system enhancement and maintenance, CSCW systems must be necessarily designed of consider about system enhancement and maintenance Also, there is a problem that coordination of cooperative work in legacy CSCW systems is developed statically from the beginning of development stage, it is difficult to support dynamic interactive cooperation in cooperative work.

In this research, we design CORBA based dynamic collaborative work system. The proposed system consist of session, shared object, access control manager, and coordinator for changing individual and group policy defined by CPDL dynamically in runtime. So this system can change coordination policy dynamically in runtime and supoort the various type of cooperative work. Also, this system provide an easy way to enhance maintain CSCW system.

요 약

분산 환경에서의 공동 작업을 위한 CSCW 시스템은 공동 저작, 화상회의 등의 분야에서 다양하게 이용된다. 그러나 공동작업 시스템들은 시스템 확장 및 유지보수에 대한 고려 없이 설계되기 때문에 시스템 확장 및 유지보수를 고려한 설계가 필요하다. 또한 기존의 공동작업 시스템들은 개발 초기에 공동작업에 대한 정책이 결정되기 때문에 공동작업 시 사용자간의 동적인 상호협력적 공동작업을 지원하지 못하는 문제점을 가지고 있다.

따라서, 이 연구에서는 분산 객체 표준인 CORBA를 기반으로 하는 CORBA 기반 동적 상호협력 공동작업 시스템을 설계하였다. 설계한 시스템은 그룹 단위 협동을 위한 세션 및 공유 객체, 접근 제어 관리자를 두어 사용자간의 공동작업을 지원하고 논리 기반의 명세 언어인 CPDL로 기술한 공동작업의 정책을 동적으로 정책을 변경 가능하게 하는 조정자로 구성된다. 따라서 공동작업 시에 CPDL로 공동작업의 정책 변경을 동적으로 변경가능 하게 하고 다양한 공동작업의 형태를 지원할 수 있다. 또한 CORBA로 시스템을 설계함으로써 시스템 확장 및 유지보수의 용이함을 제공한다.

1. 서론

네트워크의 발달과 컴퓨팅 환경의 개선으로 인한 화상 회의 시스템, 상호 협력적 원격 교육, 원격 진료, 그룹 게임 분야 등의 컴퓨터 지원 공동 작업(CSCW: Computer Supported Cooperative Working)에 대한 연구가 활발히 진행되고 있다[AVE95].

이러한 공동작업 시스템에 대한 기존 연구로는 MMConf, SharedX, Rendezvous, Diamond등이 있다[ABA93,JLA90,JPA90,MAL93]. 이러한 공동작업 시스템들은 다수의 참여자가 공동작업을 원활하게 수행할 수 있도록 다양한 지원을 하고 있으나, 시스템의 확장 및 유지보수에 대한 고려가 부족하다. 즉, 공동작업 시스템은 기존의 독립형 시스템 개발과는 달리 개발 방법, 확장 및 유지 보수 작업이 매우 복잡하기 때문에 개발자에게 많은 시간과 노력을 요구한다[OMG98]. 따라서 향후의 시스템의 확장 및 유지보수를 고려하여 시스템이 설계되어야 한다.

OMG(Object Management Group)의

CORBA(Common Object Request Broker Architecture)는 이기종 분산 환경에서의 손쉬운 응용 개발 방법을 제공하는 표준으로 응용 개발에 필요한 여러 기능들을 서비스로 정의하고 있다[OMG98, 이99].

또한, 기존의 연구들은 여러 사람들이 공동 작업을 할 때 각 개인이나 그룹에서의 조정 정책(coordination policy)들이 정적으로 개발 초기 단계에서부터 정해져서 개발되어지기 때문에 공동작업 시에 동적인 상호협력을 지원하지 못한 다는 문제점을 안고 있다.

따라서 이 연구에서는 이러한 문제점을 해결하기 위한 방법으로는 조정 정책과 실제 연산을 분리함으로써, 개인 및 그룹의 정책을 실행 시간에 동적으로 변경이 가능하도록 하고, 이종 분산 환경의 표준 플랫폼인 CORBA를 기반으로 동적 상호협력 공동작업 시스템을 설계한다. 설계할 시스템은 그룹 단위 협동을 위한 세션관리자(Session Manager) 및 공유 객체(Shared Object) 관리와 접근 제어(Access Control) 모델을 포함하는 공동 작업 계층(Collaboration Layer)과 이중 버스 구조의 상호 협조 구조를 가지는 논리 기반의 명세 언어인 CPDL를 개발자에게 제공함으로써 다중 사용자 공동 작업 환경에서 그룹 단위 정책과 개인 단위 정책을 조정할 수 있도록 정책 조정 계층(Coordination Layer)을 갖는 계층형 구조로 정의한다. 또한 계층간의 인터페이스를 독립시킴으로써 이식성이 높고, 분산 응용 개발의 표준 방식인 인터페이스 정의 언어를 사용하여 정의함으로써 구현의 유연성을 제공할 수 있다.

Ⅰ. CORBA와 기존 공동작업 시스템

이 장에서는 CORBA에 대해 고찰하고 기존 공동작업 시스템에 대해 알아본다.

2.1 OMG의 CORBA

OMG(Object Management Group)에서 제안한 CORBA(Common Object Request Broker Group)는 분산 환경에서의 객체 기술을 이용한 응용 개발 표준으로, 이기종 환경에서의 손쉬운 응용 개발 방법을 제공한다. CORBA 환경은 사용자에게 시스템의 분산과 이질성에 대한 투명성을 보장하며, 응용 개발에 필요한 여러 기능들을 서비스로 정의하고 있다. 현재 OMG에서 제안한 CORBA에 대한 표준은 2.3까지 나와 있는 상태이다[OMG98,이99].

크게 객체 통신을 담당하는 ORB(Object Request Broker)와 응용 개발에 필요한 여러 기능들을 서비스 형태로 정의한 COSS(Common Objecct Service Specification)로 구성된다. 그림 1은 CORBA의 구조를 나타낸다.

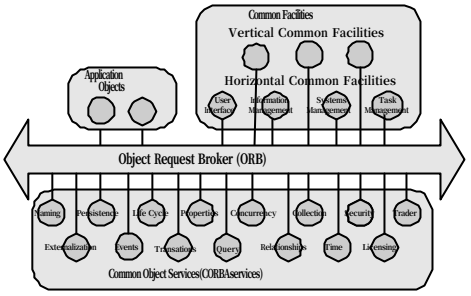

그림 1. CORBA의 구조

분산 응용 개발에 있어서도 응용 개발자는 COSS를 사용함으로써 시스템의 상호 작용성을 높이고 시스템 구현을 단순화시킬 수 있는 장점을 얻는다.

특히, 1998년 7월 OMG에서 새롭게 정의한 CORBA Telecom은 CORBA 환경에서의 분산 멀티미디어 응용들을 개발하기 위한 환경을 제공한다. CORBA Telecom은 연속된 프레임들을 정의한 흐름(flow)객체와 객체간의 흐름의 집합인 스트림(stream)객체를 정의하고 있다[OMGTE]. 오디오/비디오 데이터의 제어는 기존의 ORB를 통해서 스트림 인터페이스 조정 객체에 의해서 이루어지며, 실제 오디오/비디오의 흐름은 스트림 어댑터를 이용하여 ORB 밖에서 이루어지는 형태로 정의되어 있다.

CORBA Telecom의 정의는 멀티미디어 개발에 부적합한 CORBA 환경을 확장함으로써 기본 CORBA가 제공하는 분산 응용 개발의 이점을 분산 멀티미디어 응용에서 이용할 수 있게 되었다. 따라서, CORBA 환경에서의 분산 멀티미디어 응용 개발이 활발하게 이루어 질 것으로 예상된다.

2.2 기존 공동 작업 시스템 분석

기존의 공동작업 시스템을 설계하는 방법에는 크게 다음의 세 가지로 구분할 수 있다[AVA95,조98,ABA93,JLA90,JPA90].

중앙 집중형 공동작업 투명성(Centralized

Collaboration-transparent) 방식은 공동작업 시 사하게되는 어플리케이션의 윈도우 인터페이스를 복제함으로써 공동작업도구들을 공유하게 하는 방법이다. 다양한 출력장치에서 이벤트 스트림을 가로채고 동시에 윈도우를 구동하는 메카니즘을 사용하며 대표적인 시스템으로는 SharedX, XTV와 COMIX가 있다. 그러나 이 방식은 구현하기가 쉽다는 장점을 가지지만 사용자간의 상호작용을 원활하게 지원하지 못하며 네트워크 트래픽에 문제점을 가지게 된다[ABA93,MAL93].

중앙 집중형 공동작업 인지 방식은 다자간 소프트웨어 인스턴스가 다양한 인터페이스를 구동함으로써 다자간에 공유토록 해주는 방식으로 도구가 내부적인 발언권 제어 메카니즘을 구현함으로써 동시에 다자간 상호작용을 가능하게 한다. 그러나 이기종 분산환경에서는 하드웨어의 종속적인 특징 때문에 복잡성이 증가하고 모든 이벤트가 중앙 서버를 이용하기 때문에 네트워크 트래픽에 문제가 있다[JPA90].

복제 공동작업 인지방식은 분산 환경에서 도구의 인스턴스를 가지고 있으며 각각은 국부 인터페이스가 있고 공유 태스크를 접근하여 공동작업을 하며 입출력 일치성과 어플리케이션에 대한 구동 일치성을 보장한다. 그러나 이 방식은 도구 내부에 발언권 제어 메카니즘 구현이 가능하며 동시에 다자간 공동작업을 가능하게 하지만 공유 상태와 상호 동작 상태를 유지하는 복잡성 증가와 동시 접근 제어가 어려운 문제점을 가진다[JLA90].

2.3 정책 기술 모듈을 포함한 공동 작업 시스템

2.3.1 절차적 언어에 의한 해결방법

전통적인 CSCW 응용의 개발 방식은 C/C++와 같은 절차적 언어를 이용해서 이루어졌다. 따라서, 이러한 응용들은 응용을 개발 할 때마다 매번 반복적이고 똑같은 구현 노력을 들어야만 했다. 그 후, 설계자들은 중복 개발을 방지하기 위해 그룹웨어 시스템에서 필요한 기본적인 서비스들을 추출해서 응용 개발자들에게 GroupKit[Gree]과 MASH[Mash]등의 그룹웨어 툴킷(Toolkit)의 형태였다. 그룹웨어 툴킷은 다중 사용자 인터페이스를 만들기 위한 프로그래밍 인터페이스의 라이브러리를 제공하고 그룹 통신과 데이터 뷰를 표현과 분리하는 등의 약간의 프로그래밍 추상화를 제공하였다. 이런 방식으로 많은 구현 노력을 줄일 수 있었으나, 조정 정책은 여전히 절차적 언어로 기술되었고 구현 모듈 안에 포함되어서 만들어졌다. 이러한 방식은 여전히 구현 시나 실행 시 구현 정책을 변경할 수 있는 유연성을 제공하지 못했다. 이런 방식으로 개발된 툴들은 일반적으로 다른 툴들과 상호 작용할 수 없다. 이와 같은 이유로 CSCW 응용 사용자들은 그들의 특정 요구를 제대로 만족하는 시스템을 찾기 힘들다.

2.3.2 DCWPL

DCWPL(Describing Collaborative Work Programming Language)은 SUNY at Stony Brook에서 연구된 그룹웨어(Groupware)를 위한 규약 언어이다[DWC]. DCWPL은 프로그래머가 어떤 전통적인 프로그래밍 언어로 쓰여진 계산적인 프로그램(computational program)으로부터 분리된 제어 메커니즘을 지정함으로써 조정 프로그램(coordination program)을 정의하는 것을 허용한다. 계산적인 프로그램은 공유 가능한 요소들, 예를 들어 데이터 상태 또는 함수들의 실제적인 설명을 포함하는 반면 조정 프로그램은 사용자가 다른 팀 구성원과 이 요소들을 가지고 상호 작용할 수 있는 방법을 지정한다. DCWPL은 공유 자원들에 대한 공통 태스크의 수행을 조절하기 위한 메커니즘을 제어 및 조정하는 역할을 한다.

그러나, DCWPL에서는 참여자들을 처리하기 위한 전체 그룹에 대한 세션(Session) 생성, 세션 참여, 세션 탈퇴 등의 연산에서 제한적인 정책만 정의가 가능하다. 따라서, 그룹 내에서 참여자들 간에 발생하는 다양한 상호 작용에 대한 정의가 불가능하며, 각 참여자들의 정책을 정의하지 못하기 때문에 자율적인 그룹 활동에 대한 보장이 되지 못한다.

Ⅲ. CORBA 기반 동적 상호협력 공동작업 시스템

이 장에서는 동적으로 공동작업의 정책을 변경 가능하도록 하는 기술언어를 이용하여 동적 상호협력이 가능한 CORBA 기반 동적 상호협력 공동작업 시스템을 설계한다.

3.1 시스템 설계 개요

CORBA 기반 동적 상호협력 시스템은 그림 2와 같이 구성된다.

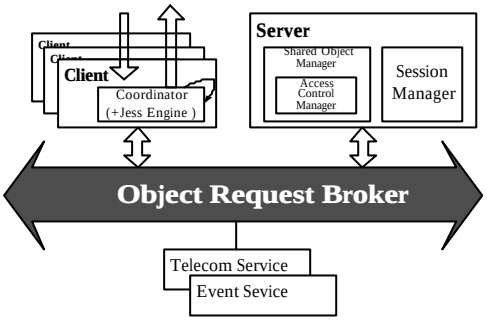

그림 2. 시스템 구성도

그림 2에서 클라이언트에서 요구된 연산은 특정한 객체에 관한 연산인 경우 바로 서버 측으로 전송되지만, 정책 변경 같은 추론을 요구하는 경우 클라이언트 측의 추론 엔진을 통해 분석되고, 결과적으로 요구되는 연산들을 요구하게 된다. 요구된 연산은 세션 측의 접근 제어 관리자를 통해 사용자 인증 과정을 거치게 되며, 인증을 거친 요구는 세션 관리자에서 역할 기반 접근 테이블에 해당 내용을 추가하고, 사용 권한을 판별하여 이벤트 통지를 통해 공유 객체를 생성하게 된다.

객체 그룹 생성 요구 시에는 공유 객체 관리자에서 객체 그룹 팩토리를 이용하여 객체 그룹을 생성하게 되고, 이후부터는 객체 생성 시에 공유 객체 관리자를 통하여 객체를 등록하고 공유하여 사용하게 된다.

3.2 조정자

공동 작업의 정책은 CPDL로 기술되고, 이렇게 기술된 정책은 조정자에 의해 관리되고 실행된다. 그림 3은 조정자의 전체적인 구성을 나타나타 내며 조정자에서 사용하는 추론 엔진으로는 Java 클래스 파일을 사용할 수 있는 Jess(Java Expert System Shell)[Jess98] 엔진을 사용한다.

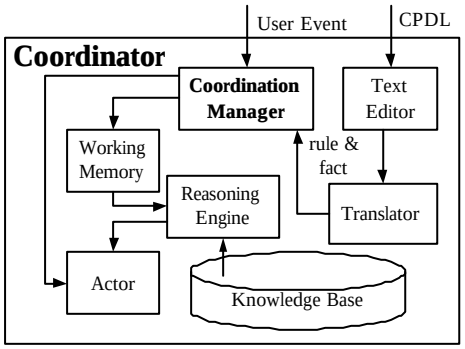

그림 3. 조정자의 구성

텍스트 편집기는 사용자가 CPDL를 기술하기 위한 환경을 제공한다. 파서는 사용자가 정의한 CPDL을 조정 관리자에서 사용할 수 있는 문법의 형태로 변환하여 주는 역할을 한다. 조정 관리자는 공동 작업에서 사용되는 전체적인 정책을 제어한다. 즉, 사용자로부터의 정책에 대한 사건을 인지하고, 이에 대해 적절한 처리를 하기 위해서 조정 관리자는 행위자가 수행할 수 있는 사건이 발생된 경우에는 사건에 대한 행동을 취하도록 행위자에게 메시지를 전달하며, 그렇지 않은 사건이 발생된 경우에는 작업 메모리에 입력된 인지된 사실을 입력하고 추론 엔진을 활성화시키도록 한다.

조정 관리자에 의해 활성화된 추론 엔진은 작업 메모리를 검사하여 발생된 사건과 현재 상태를 사실(facts)로 하고 지식 베이스에 저장된 규칙들에 의해 전방 추론(forward chaining)에 입각한 다음 행동을 결정하여 행위자에게 알리는 역할을 한다.

지식 베이스의 규칙들에 의해 사용되는 정책에 대해 발생된 사건과 현재 정책의 상태에 대한 정보를 사실로 가지고 있다. 처리된 사건은 추론 엔진에 의해 삭제되며 정책에 대한 정보는 바뀌게 된다. 지식 베이스는 공동 작업에서 사용되는 정책이 갖는 일반적인 규칙의 집합으로 구성된다. 행위자는 추론 엔진을 통해 추론된 사건들이 실행하는 실제 자바 객체의 메소드가 된다. 즉, 추론 엔진은 작업 메모리를 검사하여 발생된 사건과 현재 상태를 사실로 하고 지식 베이스에 저장된 규칙들에 의해 전방 추론에 입각해 취해지는 행동에 의해 실제 구현자가 구현한 시스템 내의 연산이 실행된다.

3.2.1 규칙의 실행

그룹에서 발생하는 사용자들의 상호 작용과 각각의 참여자들이 전달되는 이벤트에 대해서 처리하는 규칙은 CPDL로 기술된다. 조정 관리자는 각각의 CPDL로 정의한 규칙에 만족한 사실이 들어오면 실행 될 수 있도록 모든 규칙을 미리 정의하고 있으며 정의한 행위가 정의된 대상자에게 기술된 행위가 이루지게 한다.

다음의 리스트 1의 규칙들 대상은 지역, 조건에서 모드는 입력 형태, 데이터형은 REQUEST_TOKEN, 시스템의 상황은 Teaching, 수행자는 학생이라는 규칙 정의에 대한 행위와 대

상은 모든 참여자, 모드는 입력 형태, 데이터형은 CommandTeaching, 시스템의 상황은 모든 상황, 수행자는 강사이라는 규칙 정의에 대한 행위를 정의한 것이다. 이 규칙들은 시스템 상황이 Teaching 단계일 때는 학생으로부터의 토큰 요구를 거절하겠다는 규칙과 모든 시스템 상황에서 강사의 CommandTeaching 이라는 이벤트가 발생하였을 때 상황을 Teaching 단계로 변경하라는 규칙을 정의한 것이다.

리스트 1. 규칙에 의한 행위의 정의

```
RULES :   RULE :
             LOCAL : IN :-REQUEST_TOKEN
             WHEN Teaching BY Student
          = > "Deny Request"
          RULE :
            ALL : IN :-CommandTeaching
            WHEN ALL BY Lecturer
          = > "Command by Lecturer"
...
```

이러한 CPDL로 기술된 규칙들은 파서를 통해 추론 엔진에서 해석할 수 있는 형태로 변환된다. 리스트 2는 리스트 1에서 정의한 그룹 규칙이 변환되어 추론 엔진에서 처리될 수 있는 CLIPS(C Language Integrated Production System) 문법[Jess98]으로 변경된 결과를 나타낸다.

리스트 2. 리스트 1의 CPDL의 변환 결과

```
(deftemplate STAGES (slot stage))
(deftemplate RULE (slot targetlist)
                  (slot mode)
                  (slot datatype)
                  (slot WHEN)
                  (multislot By)
)
(defrule Group_RULE1
   (RULE (targetlist LOCAL)(mode IN)(datatype
         REQUEST_TOKEN)
         (WHEN Teaching)(By Student))
         (STAGES (stage Teaching))
   = > (printout t "Deny Request" crlf )
       (call RequestHandling LOCAL IN
        REQUEST_TOKEN Teaching Student
        Deny_Request))
(defrule Group_RULE2
   (RULE (targetlist ALL)(mode IN)(datatype
          CommandTeaching)
          (WHEN ALL)(By Lecturer))
          (STAGES (stage ALL)
   = > (printout t "Command by Lecture" crlf)
       (call RequestHandling ALL IN
        CommandTeaching ALL Lecturer
        Command by Lecture))
.....
```

미리 정해진 LOCAL : IN : REQUEST_TOKEN : WHEN : Student에 해당하는 규칙과 STAGES 값과 사용자가 입력한 규칙이 조건이 일치할 때 화면에 Deny Request라는 메시지를 출력하고 해당 대상자에게 시스템에서 구현자가 정의한 행위인 RequestHandling 객체 내의 DenyRequest 연산이 수행된다.

3.3 세션 관리자

세션 관리자는 새로운 그룹의 생성이나 그룹의 소멸등의 요구에 대해서 그룹 관리자를 관리하며, 그룹 관리자는 그룹 정보를 관리하고 그룹 내에서의 사용자간의 공동 작업을 지원한다. 통지 모듈은 이벤트의 정의 기능과 정의된 이벤트를 그룹에 등록하는 기능을 수행한다. 그림 4는 세션 관리자의 구성도이다.

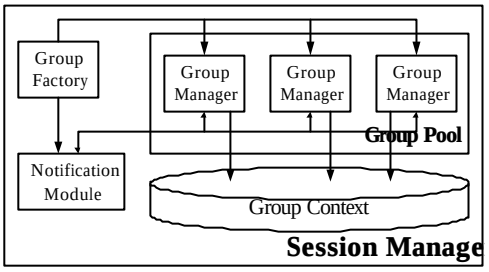

그림 4. 세션 관리자

그룹 팩토리는 새로운 그룹 요구에 대해서 그룹의 생명주기를 관리한다. 또한, 그룹 팩토리는 현재 활동 중인 그룹의 리스트를 관리하며 사용자의 정보 요구에 대해서 해당 정보를 사용자에게 제공한다. 리스트 3은 세션 관리자의 인터페이스를 OMG의 IDL(Interface Definition Language)로 기술한 것이다.

리스트 3. 세션 관리자 인터페이스

```
typedef                    seqrence<GroupInformation>
GroupInformationList;
typedef seqrence<RoleInformation> RoleList;
typedef seqrence<StageInformation> StageList;
enum Group_Attribute{PUBLIC, PRIVATE};
enum          FloorControlType{TOKEN_CONTROL,
ROUND_ROBIN, FREE}
```

```
interface GroupFactory {
GroupManager GroupCreate(in ClientID clientID, in string passwd, in string title, in RoleList rolelist, in GroupAttribute attributes, in FloorControlType controlType, in unsigned short max);
boolean GroupDestroy(in ClientID clientID, in string passwd, in GroupID groupID);
  ....
};
interface GroupManager{
boolean giveRole(in string passwd, in ClientID clinetID, in Role role );
boolean requestToken(in ClientID clinetID );
boolean giveToken(in ClinetID clientID);
Participant getParticipantList(in ClinetID );
RoleList listPossivleRoles();
  ....
};
interface NotificationModule {
boolean registerEvent( GroupID, EventID );
boolean unregisterEvent( GroupID, EventID );
};
```

사용자의 요구가 새로운 그룹의 생성인 경우에는 새로운 그룹 관리자를 생성하고, 그룹 정보(group context)를 관리할 수 있는 자료 구조를 생성한다. 그룹에 참여하고자 하는 사용자는 그룹의 ID를 이용해서 그룹 관리자의 참조를 얻을 수 있다. 공동 작업 관리자는 인터페이스를 통해서 전달되는 사용자의 요구에 따라서 세션 정보를 적절히 변경한다.

그룹 관리자는 그룹 생성 시 그룹 팩토리에 의해서 생성되며, 그룹의 상태에 대한 문맥(context)을 관리하고, 그룹 구성원들간의 상호 작용에 필요한 기능들을 수행한다. 또한, 공동 작업 관리자를 통해 전달되는 사용자의 요구를 처리한다. 그룹에 대한 정보는 그룹 문맥을 이용해서 유지된다. 그룹 문맥에는 현재 그룹에 참여하고 있는 참여자들의 정보와 현재 의장의 정보, 발언권을 가진 사용자의 위치 등을 갖는다.

그룹 관리자는 최근에 처리한 사건에 해당하는 그룹 문맥을 유지하는데, 만약 새로운 사용자의 참여나 사용자 탈퇴, 발언권 이동과 같은 경우 해당하는 문맥으로 문맥 전환이 이루어진다. 문맥 전환이 이루어지면 이벤트 전달 기능을 이용해서 해당 사실이 그룹 참여자들에게 통보된다. 역할은 그룹 의장으로부터 그룹 생성 시 정의되며, 참여자는 자신의 역할을 가지고 그룹에 참여한다. 그룹 생성 시에는 그룹 생성자가 그룹 의장의 역할을 담당하며, 그룹 활동 중간에 다른 사람에게 의장 역할을 부여할 수 있다. 또한, 그룹 관리자는 발언권 제어를 위한 기능들과 그룹내의 활동을 지원하기 위한 기능들을 제공한다.

통지 모듈은 그룹 내에서 사용할 이벤트를 정의하고, 그룹별로 이벤트를 등록하고 해제할 수 있는 기능을 제공한다. 통지 모듈은 그룹 생성시 그룹에서 사용할 이벤트의 종류를 정의할 수 있도록 하고, 이벤트 발생시 이를 이벤트 처리 기능을 이용해서 이벤트를 이벤트 소비자에게 전달할 수 있게 한다.

3.4 공유 객체 관리자

공유 객체 관리자는 접근 제어 관리자와 통합하여 설계한다. 사용자 권한과 객체에 대한 접근 권한을 일치시켜 사용자 역할에 의해 할당하고, 할당된 역할은 새로운 그룹을 생성하거나 객체 그룹을 생성할 때 상속되어 사용할 수 있다.

3.4.1 공유 객체 관리자

그림 5는 공유 객체 관리자의 구성을 나타낸다.

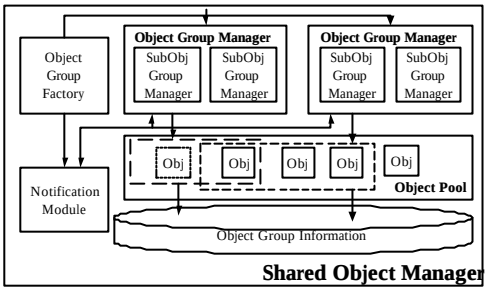

그림 5. 공유 객체 관리자

공유 객체 관리자는 참여자에게 보이는 외부와의 통신을 위한 인터페이스와 객체를 그룹으로 묶어 객체의 생명 주기를 담당하는 객체 그룹 관리자(Object Group Manager), 각각의 객체 그룹을 담당할 객체 그룹 팩토리(Object Group Factory), 공유 객체 관리자내의 변화에 대한 통지 기능을 담당하는 통지 모듈과 각각의 객체 그룹에 대한 정보를 관리하는 객체 그룹 정보 데이터베이스로 구성된다.

공유 객체 관리자는 새로운 객체 그룹의 생성이나 객체 그룹의 소멸등의 요구에 대해서 객체 그룹 팩토리를 관리하며, 객체 그룹 관리자는 등록되는 객체에 대해 세션에서 정의한 역할에 따라

서브 그룹으로 관리한다. 변화 통지자는 공유 객체 관리자내의 변화를 참여자들에게 통지해주는 기능을 담당한다. 리스트 4는 공유 객체 관리자 인터페이스를 OMG의 IDL로 기술한 것이다.

리스트 4. 객체 그룹 팩토리 인터페이스

```
interface ObjectGroup_Factory{
ObjectGroupManager CreateObjectGroup(in ClientID
clientID, in string passwd, in GroupID groupID);
boolean DestroyObjectGroup( in ClientID clientID, in
string passwd, in GroupID groupID);
  ...
};
interface ObjectGroupManager {
SubObjectGroupManager createSubObjectGroup (in
ClientID clientID,in string passwd, in Role role );
boolean DeleteSubObjectGroup(in ClientID clientID, in
string passwd, in Role role);
SubObjectGroupManager SearchSubObjectGroup(in Role
clientRole);
};
interface SubObjectGroupManager {
attribute string SubObjectGroupID;
boolean InsertObject (in ClientID clientID, in string
passwd, in ObjectID objectID, in string type, in Role role
);
boolean DeleteObject( in ClientID clientID, in string
passwd, in ObjectID objectID, in Role role);
boolean updateObject (in ClientID clientID, in string
passwd, in ObjectID objectID, in string type, in Role
role)
  ....
};
interface Concurrency_Manager{
boolean requestLock(in ObjectID objectID, in ClientID
clientID, in LockType lockType);
boolean requestUnlock(in string ObjectID, in string
ClientID);
};
```

서브 그룹의 생명주기를 관리하기 위해 객체 그룹 관리자는 역할마다의 서브 그룹을 생성하는 연산과 서브 그룹에 객체를 등록/삭제하는 연산 그리고 참여자에게 자신의 역할의 서브 객체 그룹에 등록된 객체의 정보를 제공하는 연산 등으로 구성된다.

객체를 사용하고자 하는 참여자는 서브 그룹 관리자에서 제공하는 연산을 사용하여 해당 객체의 참조 값을 반환 받는데 객체에 대한 요구를 이전에 동시성 관리자에서 제공하는 잠금 연산을 이용하여 TRUE값을 받아야만 객체를 이용할 수 있다. 제공하는 연산은 해당 객체가 다른 사용자에 의해 사용될 때는 false 값을 반환함으로써 객체의 동시성을 유지한다. 또한 객체 그룹에 등록되거나 삭제되고 역할의 변화에 따른 객체 그룹 정보의 변화는 변화 통지자를 통해 각각의 참여자에게 정보를 제공하게 된다. 알고리즘 1은 동시성 관리자에서 잠금 연산을 수행하는 알고리즘이다.

알고리즘 1. 특정 객체에 대한 잠금의 수행

```
Begin Method
  Check requestLockType,
  Check Lock_Type of Object Currently,
  If CurrentLock of Object is not NULL then
    Case Lock_Type of Object Currently,
      READ_LOCK:
        Case requestLockType:
          READ_LOCK, SHARED_LOCK:
                                  return TRUE;
          WRITE_LOCK: return FALSE;
        End of Case
      WRITE_LOCK:
        Case requestLockType:
         READ_LOCK,SHARED_LOCK,
         WRITE_LOCK:
                return FALSE;
        End of Case
      SHARED_LOCK:
        Case requestLockType:
          READ_LOCK, SHARED_LOCK:
                        return TRUE
          WRITE_LOCK:
             Create Replicated Object
                          for current Object,
             Notify Replicated Object Information
                                        to Client,
             Return TRUE;
          End of Case
      End of Case
  End of If
ELSE
  Change Lock_Type of Object into requestLockType
              and ObjectInformation;
  End of If
End of Method
```

3.4.2 접근 제어 관리자

접근 제어 관리자는 접근 제어 관리자를 관리하기 위한 인터페이스와 사용자로부터 전달되는 접근 요청에 대해서 접근 제어 기능을 제공하기 위한 접근 결정기로 구성된다. 또한, 인터페이스와 역할별 접근 가능한 객체의 정보를 관리한다.

접근 결정기는 역할별로 정의되어 있는 접근

제어 정보를 이용해서 사용자가 특정 객체에 대한 권한이 있는지를 조사하는 기능을 수행하는 접근 제어 관리자의 핵심적인 부분이다.

리스트 5. 접근 제어 관리자의 관리 인터페이스

```
interface ManagerAdmin {
boolean insertControlData(in GroupID groupID, in Role role, in Object obj, in Right right);
boolean deleteControlData( in GroupID groupID, in Role role, in Object obj, in Right right);
boolean modifyRight(in GroupID groupID, in Role role, in Object obj, in Right right);
boolean createGroupData(in GroupID groupID);
boolean deleteGroupData(in GroupID groupID);
boolean inherit(in GroupID groupID, in Role parentRole, in Role childRole );
};
interface AccessControl {
boolean isAllowed(in GroupID groupID, in Role role,in Object obj, in Right right);
ControlDataList retrieveGroupData(in GroupID groupID);
ControlDataList retrieveRoleData(in GroupID groupID, in Role role);
ColtrolDataList showWriteEnable(in GroupID groupID, in Role role);
};
```

Ⅳ. 실험 및 평가

이 장에서는 설계한 CORBA 기반 동적 상호협력 공동작업 시스템을 구현하고, 구현한 응용의 수행 과정을 통해 설계한 시스템의 각 모듈의 기능을 평가한다. 또한, 그룹에 참여하는 사용자들이 CPDL로 정의된 그룹 정책과 개인 정책에 따라서 상호작용이 정확히 발생하는지를 보인다.

4.1 개발 환경 및 교육 시스템

실험 환경은 1명의 강사와 2명의 학생 그리고 1명의 청강생으로 이루어진 수업에서 강사가 그룹의 형태와 그룹 정책을 결정하게 한다.

전체적인 교육 시스템은 화이트보드와 실시간 스트림을 이용한 화상회의 시스템으로 구성된다. 화상회의 시스템에서의 영상/음성 데이터와 화이트보드의 영상 데이터는 그룹 정책에 따라서 흐름이 제어된다.

동적 상호협력 공동작업 시스템을 이용한 교육 시스템은 Microsoft Windows NT 4.0과 Windows98에서 동작하며, 클라이언트/서버간 통신을 위해서는 100base-TX ethernet 상에서 TCP/IP 프로토콜을 사용한다. 또한, CORBA 기반 개발 환경과의 통신을 위한 미들웨어로 IONA사의 OrbixWeb 3.1과 Orbix 3.0을 이용하고 멀티미디어 통신을 위해 OrbixMX 베타버전을 사용한다.

4.2 실험 시나리오

교육이 이루어지기 위해서는 먼저 강사가 세션 관리자를 통해서 그룹을 생성해야 한다. 그룹을 생성하기 위해서는 여러 그룹 정보가 필요하고 그룹 정책이 결정되어야 한다. 그룹 정책이 결정되고 그룹이 생성되면, 학생들과 청강생이 그룹에 참여하고 자신의 정책을 결정한다.

강사는 먼저 Overview of CSCW라는 강의를 개설하기 위해서 세션 관리자를 통해서 그룹을 생성한다. Overview of CSCW 그룹에서는 강사, 학생, 청강생의 3가지 역할을 정의하고, 수업의 단계를 강의, 질문, 토론, 보고 등의 4단계로 정의한다.

그룹에서 사용할 객체는 강사에 의해서 참고가 되는 Textbook1과 강사에 의해서 편집되고 학생에 의해서 열람되는 Textbook2, 그리고 비디오 스트림 데이터의 상호 교환을 위한 스트림 객체와 오디오 스트림 데이터의 상호 교환을 위한 스트림 객체, 그리고 화이트보드를 위한 스트림 객체로 결정한다. 수업에 참여하는 참여자의 수는 강사 1명, 학생 2명, 청강생 1명으로 제한하고 그룹의 속성은 암호를 통해서 접근할 수 있는 보호 모드로 설정한다. 초기 발언권 제어 형태는 그룹 의장에 의해서 토큰을 이용해서 제어하고, 그룹에서 사용할 이벤트는 수업의 단계의 변경을 위한 이벤트와 토론의 결과를 강사에게 전달하기 위해서 정의된다.

수업의 단계에서 강의 단계인 경우에는 학생들의 발언권 요청을 무시하고 토론 단계에서 학생들이 토론 결과를 전달하겠다고 하는 경우에는 이를 강사가 수신할 수 있게 한다. 또한, 강의 중에는 학생이나 청강생으로부터 영상 데이터는 전달받을 수 있으나 음성 데이터와 화이트보드를 통한 데이터 전달은 원활한 강의를 위해서 규제한다.

4.2.1 그룹 정책 기술

그룹을 생성하기 위해서 필요한 정보는 그룹 이름, 그룹에서 사용할 역할들의 종류, 그룹의 작업 단계의 종류, 그룹에 참여할 총 참여자의 수와 역할별 수, 그룹의 사용자들에 의해서 사용될 객체들의 정보, 발언권 제어 형태, 그룹의 속성과 발언

권 제어 형태, 그룹에서 특별하게 사용할 이벤트의 정의, 그리고 그룹 활동에서의 규칙들로 구성된다. 이와 같은 그룹 형태와 그룹 정책은 이 연구에서 설계한 CPDL을 이용해서 리스트 6과 같이 기술한다.

리스트 6. 교육 시스템에서의 그룹 정책

```
Classroom : Overview of CSCW
 ROLES : Lecturer Student Attendant
 STAGES : Teaching Question Discussion Reporting
 OBJECTS :
    OBJECT : material1 Textbook1
             USING_BY : Lecturer R
    OBJECT : material2 Textbook2
             USING_BY : Lecturer W
    OBJECT : material3 Textbook2
             USING_BY : Student R
    OBJECT : video Stream USING_BY : ALL W
    OBJECT : sound Stream USING_BY : ALL W
    OBJECT : whiteboard Stream
             USING_BY : ALL W
 P_N : 4
    Lecture => 1
    Student => 2
    Attendant => 1
 PROTECTED : $as!123
 TOKEN_CONTROL
 EVENTS :
    EVENT CommandTeaching
       SRC : Lecturer
       DST : Student Attendant
       FILEDS :
          FIELD : Content
    EVENT ReportResult
       SRC : Student
       DST : Student
       FILEDS :
          FIELD : Content
  RULES :
     RULE :
        LOCAL : IN :- REQUEST_TOKEN
                WHEN Teaching BY Student
       => Deny Request
     RULE :
        Lecture : IN :- ReportResult
                  WHEN Discussion BY Student
                => Report Result
     RULE :
        ALL : IN :- CommandTeaching
              WHEN ALL BY Lecturer
              => Command by Lecture
     RULE :
        LOCAL : IN :- sound
                WHEN Teaching BY Student Attendant
                => Prohibit Speech
```

4.2.2 그룹 생성

리스트 6에서 기술된 CPDL은 파서를 통해서 그룹 생성을 위해 필요한 정보들로 변화된다. 그룹 생성을 위해서 필요한 정보는 세션 관리자와 공유객체 관리자 그리고 멀티미디어 통신에 전달될 데이터와 지식 베이스에 저장되어서 그룹 정책 관리에 사용될 규칙들로 구성된다. 이때, 파서는 CPDL 형태를 CLIPS 형태로 변환하기 위해서 리스트 7과 같은 템플릿을 사용한다.

리스트 7. CPDL을 CLIPS로 변환하기 위한 템플릿

```
(deftemplate ROLELIST (multislot rolelist))
(deftemplate STAGELIST (multislot stagelist))
(deftemplate OBJECTLIST (multislot objectlist))
(deftemplate EVENTLIST (multislot eventlist))
(deftemplate STAGES (slot (stage)))
(deftemplate RULE (slot targetlist
                        (slot mode)
                        (slot datatype)
                        (slot WHEN)
                        (multislot by)
)
```

파서에 의해서 변형된 CPDL의 내용은 실제 세션 관리자와 공유 객체 관리자, 멀티미디어 통신 관리자에 전달되어서 그룹 생성에 사용된다. 이를 위해서 자바 객체 GroupManager를 생성하였고, GroupManager는 그룹 생성에 필요한 정보가 입력되면 그룹 생성을 위해서 제공되는 세션 관리자, 공유 객체 관리자, 멀티미디어 통신 관리자의 적절한 메소드를 호출한다. 리스트 8은 CPDL로 정의한 규칙들을 실제 자바 메소드와 연결하기 위해서 파서에 의해서 생성된 규칙들이다. 이 규칙은 Jess의 지식 베이스에 들어가서 그룹 활동 중에 발생하는 여러 사건에 대해서 적합한 규칙이 적용된다.

리스트 8. CLIPS 형태로 변경된 규칙

```
(defrule Group_RULE1
   (RULE (targetlist LOCAL)(mode IN)(datatype
REQUEST_TOKEN)(WHEN Teaching)(By Student))
(STAGES (stage Teaching))
   = > (printout t "Deny Request" crlf )(call
RequestHandling LOCAL IN REQUEST_TOKEN
Teaching Student Deny_Request))
(defrule Group_RULE2
   (RULE (targetlist ALL)(mode IN)(datatype
```

CommandTeaching)(WHEN ALL)(By Lecturer))(STAGES (stage ALL)
= > (printout t "Command by Lecture" crlf)(call RequestHandling ALL IN CommandTeaching ALL Lecturer Command by Lecture))
(defrule Group_RULE3
(RULE (targetlist LOCAL)(mode IN)(datatype sound)(WHEN Teaching)(By Student Attendant))(STAGES (stage Teaching)
= > (printout t "Prohibit sound" crlf)(call RequestHandling LOCAL IN sound Teaching Student Attendant Prohibit sound))

그림 5은 세션 관리자가 GroupManager의 그룹 생성 요청을 처리하는 과정을 보여준다. 사용자의 새로운 그룹 생성 요구에 의해서 그룹 팩토리는 그룹 관리자와 그룹 관리자에서 관리하는 그룹 문맥을 생성하고, 이벤트 데몬을 통해서 다른 사용자들에게 그룹 생성 사건을 알린다.

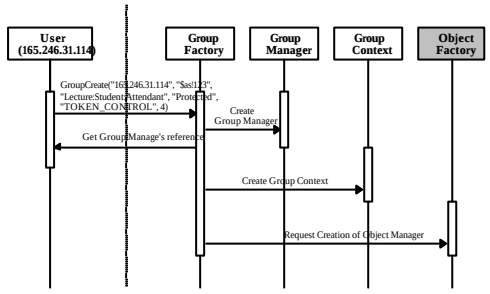

그림 6. 새로운 그룹 생성

그림 7은 그룹의 의장인 강사에게 제공되는 인터페이스이다. 그룹에 참여하는 사용자들의 역할 부여와 그룹의 상태 변경 등을 가능하게 한다. GroupManager 자바 객체는 그룹 생성에 대한 요구를 받으면 앞서 설명한 세션 관리자의 그룹 생성을 위해서 제공되는 메소드를 호출하며 그룹에서 사용할 그룹 정보를 생성한다. 또한 세션 관리자는 그룹에서 사용할 객체들을 관리하기 위해 공유 객체 관리자를 통해 역할별로 관리되는 객체 그룹을 생성하게 된다.

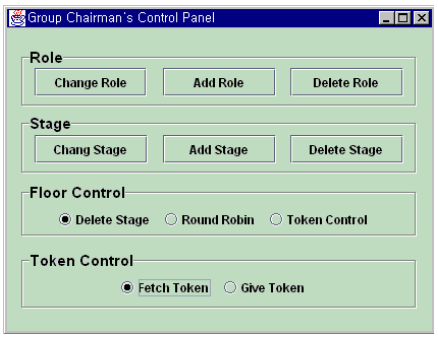

그림 7. 그룹 의장 제어 패널

그림 8은 CPDL로 정의한 공유 객체인 material1, material2 등을 공유 객체 관리자에 등록하기 위해 세션으로부터 객체 그룹의 요청을 받아 역할 기반 객체 그룹을 생성하는 단계를 보여주고 있다.

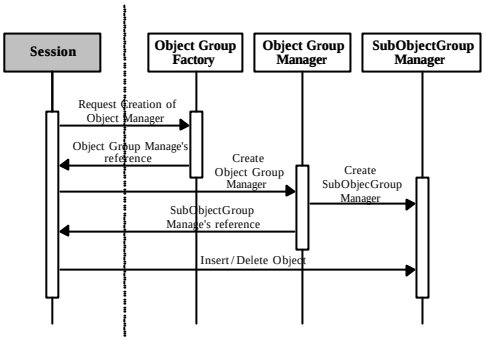

그림 8. 객체 그룹의 생성

세션으로부터 객체 그룹 생성에 대한 요청을 받는 객체 그룹 팩토리는 세션마다 공유 객체를 관리하는 객체 그룹 관리자를 생성하게 된다. 또한 객체 그룹 관리자는 역할 별로 관리되는 서브 객체 그룹을 생성하기 위해 사용되는 객체인 material1, material2 등에 대한 정보는 CPDL상에서 기술된 정보를 이용한다. 등록 시에 사용되는 정보는 CPDL에 기술한 객체의 객체 식별자, 클래스 식별자, 역할, 잠금 정보이다.

그림 9는 실제 CPDL로 기술한 공유 객체의 정보가 공유 객체 관리자의 자료구조로 관리되는 지를 보여주고 있다.

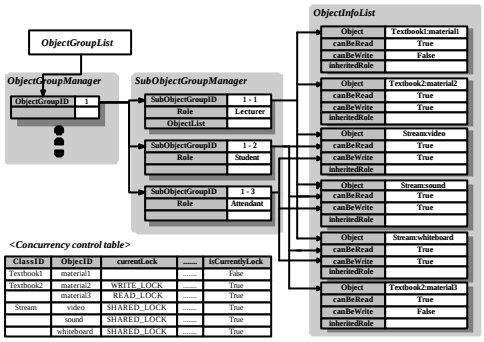

그림 9. 공유 객체 관리자에서 관리되는 자료구조

그림에서 보듯이 객체 그룹 관리자가 서브객체그룹관리자와 연결되며, 서브 객체 그룹 관리자는 역할에 기반을 두고 공유 객체들을 관리한다. 동시성 제어 테이블(Concurrency control table)은 동시성 제어 관리가 어떻게 이루어지는가를 보여준다.

4.2.3 학생 참여

교육에 참여하는 학생은 현재 개설되어 있는 강의의 리스트를 이용해서 참여할 강의를 결정한다. 그림 10은 학생이 강의에 참여할 수 있게 하는 인터페이스이다. 이 인터페이스를 통해서 그룹의 리스트를 보여주고, 그룹의 등록, 해제, 참여, 탈퇴 등의 기능을 수행할 수 있는 기능과 그룹 정보를 변경하거나 질의할 수 있는 기능을 제공한다.

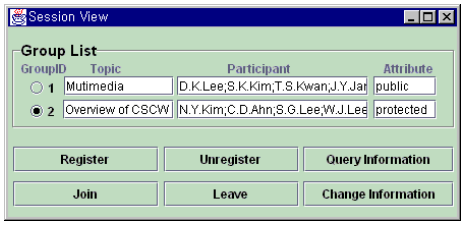

그림 10. 세션 뷰

학생이 2번 그룹을 선택하고 Join 버튼을 클릭하면 세션 관리자에 전달되어서 학생을 그룹에 참여시키고, 그룹 문맥을 변경시킨다.

학생은 현재 개설되어 있는 과목을 조사한 후, 원하는 과목이 진행되는 그룹에 참여해서 학습한다. 학생은 일반 학생과 청강생으로 구분되며, 일반 학생은 발언권을 가질 수 있는 대신 청강생은 발언권을 가질 수 없는 차이가 있다. 일반 학생은 그룹 활동에서 자신의 정책을 가질 수 있으며, 그룹 의장이 결정한 그룹 정책과 함께 전체적인 그룹 활동을 정의한다.

학생 A는 강사로부터 단계 변화 이벤트가 전달되는 경우 Change Mode 메시지를 발생시켜 전체 응용의 단계를 변화시킨다. 또한 질문 시간에 학생이나 청강생이 화이트보드를 통해서 데이터를 전달하는 경우 이를 화면에 표시하지 않도록 하고 토론의 결과를 보고하는 경우에도 자신의 화이트보드에 반영하지 않으려 하는 경우 리스트 와 같이 CPDL로 기술한다.

```
RULES :
RULE : LOCAL : IN :- CommandTeaching WHEN ALL
BY Lecturer => Change Mode
RULE : LOCAL : IN :- whiteboard WHEN Question
BY Student Attendant => Prohibit whiteboard
RULE : LOCAL : IN :- whiteboard WHEN Report BY
Student Attendant => Prohibit whiteboard
```

학생 B는 학생 A와 마찬가지로 모든 단계에서 단계 변화 이벤트가 전달되는 경우 응용의 단계를 변경한다. 또한, 질문 시간에 학생이나 청강생으로부터 음성 데이터가 전달되는 경우 이를 전달 받지 않으려고 하는 경우 리스트와 같이 CPDL로 기술한다.

```
RULES :
RULE : LOCAL : IN :- CommandTeaching WHEN ALL
BY Lecturer => Change Mode
RULE : LOCAL : IN :- sound WHEN QUESTION BY
Student Attendant => Prohibit sound
```

이렇게 정의된 학생 A와 학생 B의 CPDL은 강사가 정의한 그룹 CPDL과 같은 방식으로 템플릿을 이용해서 CLIPS로 변환되고 지식 베이스에 저장된다. 추론엔진은 정의된 조건이 만족하는지를 검사해서 만족하는 경우 해당 메시지를 전달하고, 해당 메시지에 연결된 자바 객체가 동작하도록 한다.

그림 11은 강사와 학생들로 구성된 학습에서의 그룹 정보를 보여준다. 그룹 정보에는 그룹의 제목, 역할과 역할별 사용자와 사용자 수, 그룹에서 사용되는 단계, 속성, 발언권 제어 모드, 발언권 소지자 등의 정보를 보여준다.

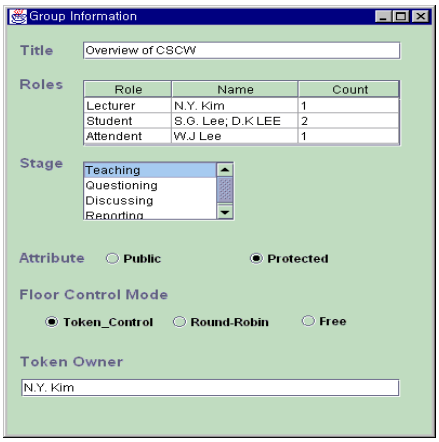

그림 11. 그룹 정보

한명의 강사와 두 명의 학생, 한명의 청강생으로 이루어진 교육 시스템의 실행 화면이다. 교육 시스템은 화상 회의 시스템과 화이트보드로 구성되며, 화상 회의에는 전체 사용자들을 비디오 스트림과 오디오 스트림을 통해서 연결하고 있으며, 화이트보드도 스트림을 통해서 연결되어 있다. 비디오 스트림인 경우에는 발언권에 관계없이 전달될 수 있으나, 오디오 데이터와 화이트보드인 경우에는 발언권이 있는 사용자의 데이터가 전달된다.

4.2.4 교수의 정책 변경

한번 정책이 결정된 후 새로운 정책을 추가하는 경우에는 전체 정책을 다시 컴파일하지 않고 새로운 규칙만 컴파일해서 지식 베이스에 추가하면 새로운 정책을 수행할 수 있다. 예를 들어서 그룹 단계 중에서 Teaching 단계가 아닌 경우에는 다른 사용자의 접근을 금지하는 경우에는 다음과 같이 CPDL을 추가해주면 된다.

```
RULE : LOCAL : IN :- JOIN WHEN Question BY
Student Attendant =>Prohibit JOIN
RULE : LOCAL : IN :- JOIN WHEN Discussion BY
Student Attendant =>Prohibit JOIN
RULE : LOCAL : IN :- JOIN WHEN Reporting BY
Student Attendant =>Prohibit JOIN
```

또한 공유 객체에 대한 정책의 변경도 변경된 공유 객체에 대한 정책을 CPDL로 기술하여 지식 베이스에 변경하여 추가하여 주면 된다. 예를 들어 다음과 같이 공유 객체의 정보를 변경하는 CPDL을 지식베이스에 추가했다고 한다면 해당하는 공유 객체의 정보는 그림 12와 같이 변경된다.

```
OBJECTS :
OBJECT : material1 Textbook1 USING_BY : Lecturer W
OBJECT : sound Stream USING_BY : Lecturer W
OBJECT : whiteboard Stream USING_BY : Lecturer W
```

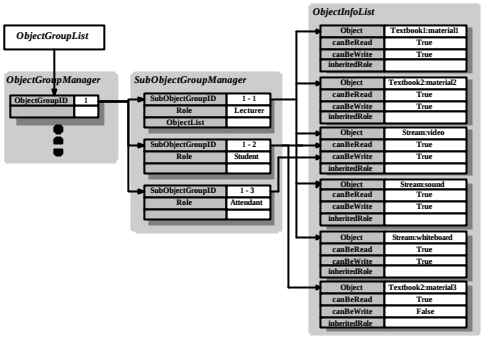

그림 12. 변경된 공유 객체에 대한 자료구조

그림 12에서 보듯이 앞서 CPDL 정의에 의해서 생성된 그림 9의 자료구조가 역할마다의 사용할 수 있는 객체의 정보가 변경되었음을 보여준다.

4.2.5 이벤트 처리

CPDL로 정의된 정책이 파싱되면 실제 응용 단계에서의 사용자의 행위를 기다리게 된다. CPDL로 기술된 정책들이 작업 메모리와 추론엔진에서 올바르게 동작하기 위해서는 응용에서 발생하는 사용자의 행위가 발생할 작업단계와 어떠한 기능을 요구하고 그 요구의 방향, 행위자의 역할이 fact로 변환된다. 이때 해당하는 규칙을 있을 경우에는 규칙에서 정의한 행위를 수행하는데 실제로 어떠한 이벤트가 발생하였는지를 검사하기 위해 자바 객체인 RequestHandling 객체를 호출한다. 호출되는 함수인 RequestHandling 자바 객체는 실제 해당기능을 수행하는 객체의 메소드를 실행시키는 역할을 담당한다. 규칙이 만족하여 RequestHandling 자바 객체를 호출할 때는 모든 규칙을 만족시킨 템플릿 내에 데이터값을 모두 전달하여 해당하는 기능을 수행하는 메소드가 무엇인지를 판단할 수 있는 기준으로 사용되게 한다.

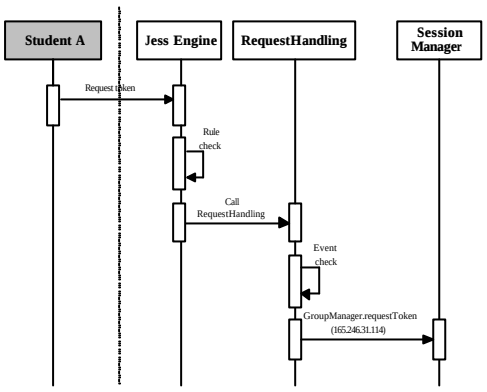

그림 13. 토큰 요구에 대한 이벤트 처리 과정

그림 13에서와 같이 학생 A가 발언권을 요구할 때 Jess 엔진에 의해 규칙이 검사되고 CPDL에서 정의한 규칙인 LOCAL : IN :- REQUEST_TOKEN WHEN Teaching BY Student에 알맞은 fact일 경우 RequestHandling 객체를 호출하며 RequestHandling 객체는 세션 관리자의 GroupManager.requestToken() 메소드를 호출하게 된다.

그림 14는 CPDL정의한 규칙에 위반되는 이벤트가 발생하였을 경우 이를 처리하는 과정을 보여주는 그림이다.

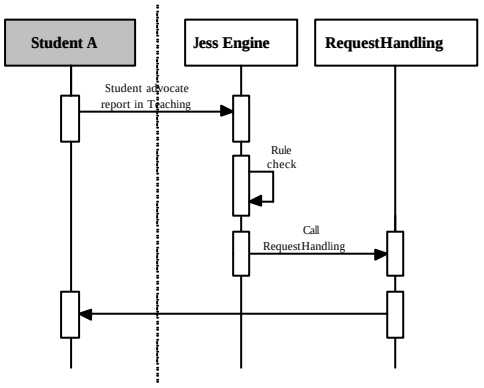

그림 14. 규칙에 위반되는 이벤트의 처리

그림 14는 학생 A가 Teaching 단계에서 레포트를 제출하는 이벤트를 처리하는 과정이다. 그러나 CPDL 정의에서 레포트는 Discussion 단계에서만 받을 수 있다는 규칙인 Lecture : IN :- ReportResult WHEN Discussion BY Student를 정의하였기 때문에 학생 A가 발생한 이벤트는 RequestHandling 객체에 의해 규칙이 위반사실을 통보받는다.

이 실험에서 제안한 시스템을 이용하여 CORBA 기반 동적 상호협력 교육시스템을 구현하였다. 구현된 교육 시스템을 특정 시나리오를 CPDL로 적절히 기술할 수 있었으며, CPDL을 CLIPS 형태로 변경할 수 있었다. CLIPS 형태로 변경된 정책은 지식 베이스에 저장되어서 추론엔진에 의해서 조건을 조사해서 조건이 만족하는 경우 특정 자바 객체의 메소드에 전달함으로써 원하는 행위를 수행할 수 있게 하였다.

따라서 기존의 공동작업 시스템과 비교해서 동적으로 정의되는 정책을 응용의 변화 없이 적용할 수 있는 공동작업 시스템을 설계할 수 있었다. 기존의 공동작업 시스템은 동적으로 변경하는 정책에 대해서 응용을 다시 개발하거나 정책을 변경해서 응용을 다시 컴파일해야 하는 문제점이 존재했다. 따라서, 동적으로 정의되는 정책이 응용의 변화 없이 적용될 수 있는지는 기존의 공동작업 시스템과 비교 평가에서 중요한 기준이 된다.

C나 C++로 응용 프로그램내에 정책 루틴이 구현되어 있는 경우 정책 변경이 발생하면 응용 프로그램의 정책 루틴을 다시 작성해서 컴파일해야 한다. 그러나, 응용 구현과 정책 기술을 분리함으로써 사용자가 그룹 정책이나 개인 정책을 변경하는 경우 응용 프로그램의 수정 없이 원하는 정책을 수행하는 공동작업 시스템을 구현할 수 있었다.

Ⅴ. 결론

이 연구에서는 CORBA 기반 동적 상호협력 공동작업 시스템을 설계하였다. 설계한 시스템은 공동작업의 조정 정책을 시스템 내에 하드코드화하지 않고, 실행 시간 중에 동적으로 조정할 수 있게 시스템과 조정 정책을 분리함으로써, 유연성을 얻을 수 있다. 이것은 그룹 정책과 개인 정책의 동적인 대응으로 다양한 응용에 적용이 가능하고, 단순한 정책 기술 언어를 제공하여 실시간 다중 사용자 공동 작업 환경에서 유연성을 제공한다.

또한 CORBA를 기반으로 설계되었기 때문에 시스템 개발자에게 시스템 확장과 유지보수의 용이성을 제공할 수 있다.

이 연구의 결과를 이용하여 동적 조정이 가능한 다양한 공동작업 시스템을 개발하기 위한 프레임워크로의 활용이 가능할 것이다.

※ 참고 문헌

[AVE95]A. Versey and Ajay Paul Sravana, "CASE as Collaborative Support Technologies," *Communication of the ACM*, pp.83-94, Jan. 1995

[OMG98]OMG, *The Common Object Servece : Architecture and Specification Revision 2.3*, OMG Document, 1998.

[OMGTE]OMG, CORBA Telecom Specification, OMG Document, 1998

[조98]조성빈, 김진석," 소프트웨어 개발 향상성을 위한 공동작업 플랫폼의 설계," 한국정보처리학회논문지, Vol.5, No.1,Jan. 1998

[ABA93]A.Babadi, "COMIX : A Tool to Share X Applications," Proc. Second Workshop on Enabling Technologies : Infrastructure for Collabarative Enterprises, pp.192 - 196,1993

[MAL93]M.Altenhifen,J.Dittrich,R.Hammerschmidt,T.Kappener,"The BERKOM Multimedia Collaboration Service," Proc. ACM Conference on Multimedia, pp.457-463,1993

[JPA90]J.Patterson,R.Hill,S.Rohall,and M.Meeks, "Rendezous:An Architecture for Syncronous Multi-User Applications," Proc. ACM Conference om Computer-Supported Copperative Work, pp. 317-328, 1990

[JLA90]J.Lauwers,T.Joseph,K.Lantz,and A.Romanov,"Replicated Architectures for Shared Window Systems:A Critique," Proc. ACM Conference on Office Information Systems, pp.249-260, 1990

[Jess98]Ernest J. Friedman-Hill, "Jess, The Java Expert System Shell," http://herzberg.ca.sandia.gov/jess/README.html, 1998

[Gree]Greenberg,S,http://www.cpsc.ucalgary.ca/grouplab/papers/index.html

[Mash]The MASH Research Group, http://mash.cs.berkeley.edu/mash/pubs/index.html

[DWC]"Digital Storage Media Command & Control," ISO/IEC JTCI/SC29/WG11 N1100 revl, p.5, Nov.1995

[이99] 이세훈, 황창종, *Inside CORBA3 프로그래밍*, 도서출판 대림, 1999.4.

이세훈

1985년 인하대학교 전자계산학과 졸업(이학사)

1987년 인하대학교 대학원 전자계산학과 졸업(이학석사)

1996년 인하대학교 대학원 전자계산공학과 졸업(공학박사)

1987~1990년 해병대 전산실 분석장교

1991~1993년 비트컴퓨터 기술연구소 선임연구원

1999~현재 (주)토마토아이 기술고문, 세이월드 기술고문

1999.6 한국멀티미디어기술사

1993~현재 인하공업전문대학 전자계산기과 부교수

관심분야 : 분산객체컴퓨팅, 멀티미디어, 소프트웨어공학, 원격교육

* 본 연구는 인하공업전문대학 산업기술연구소 99년도 교내 연구비 지원에 의하여 수행되었음.