다중사용자 인터페이스용 한글 에이전트

김 상 욱, 안 춘 근, 진 윤 숙
경북대학교 컴퓨터과학과, 컴퓨터언어연구실

A Hangul Agent for Multiuser Interface

Sangwook Kim, Choonkun Ahn, Yunsook Jin
Computer Languages Laboratory
Computer Science Department, Kyungpook National University

요 약

고속 통신망의 발달은 CSCW 의 연구를 초래했다. 본 연구에서는 CSCW 를 위한 다중 사용자용 인터페이스를 지원하는 한글 에이전트 설계에 대하여 설명한다. 본 연구의 다중 사용자용 인터페이스는 우리 글인 "한글" 사용자의 개념적 모델과 객체지향 프로그래밍 언어에 의하여 지원받는 구조 사이에서의 동질 구조를 갖도록 한다. 즉, 한글 에이전트는 인터페이스를 공유할 수 있도록 하여 모든 데이타를 개념적 모델에 의한 객체로 동시, 공유 처리한다. 본 연구의 한글 에이전트는 객체지향 방법으로 설계되어 인간의 개념 모델에 더욱 가까운 공동작업 환경을 제시함으로써, 원격지 회의, 실시간 강의 시스템 등에 응용된다.

1. 서론

지금까지의 한글 정보 처리는 대부분 단일 사용자용 인터페이스를 사용하여 워드프로세싱, 기계 번역, 문서 편집, 정보 검색등이 이루어 졌었다. 그러나 오늘날에는 공동작업을 요구하는 시대가 되었다[1-11].

멀티미디어와 컴퓨터 네트워크 기술이 발전하여, CSCW 연구가 가속화 되었다[3, 6, 12]. 이러한 CSCW 기술은 멀티미디어 정보를 처리하는 공동작업을 발전시켰으며, 다중사용자 인터페이스 기술을 야기시켰다[4,10]. 이러한 연구에서 중요한 것은 실제의 공동작업에 참여하는 것 같은 공동작업 환경을 제공하는 것이다[1,2,8]. 바로 인터페이스와 통신에 의해서 제공되어진다[12].

본 연구의 목적은 다중 사용자의 각 개념적 모델과 동질 구조로 대응되는 객체지향 다중사용자 인터페이스에서 "한글 에이전트(Hangul Agent)"를 설계, 구현하는 것이다. 이 한글 에이전트의 각 사용자는 객체 지향 인터페이스에서 모든 데이타를 개념적 모델에 따른 객체로 처리한다[1,2].

공동작업을 위한 원시적인 방법으로 E-mail[15], Talking[15] 시스템을 들 수 있는데, 이것들은 공동작업보다는 기본 적인 통신 수단에 불과하다. E-mail 시스템은 비동기 형 통신 수단으로 실시간 공동작업을 할 수 없으며, Talking 시스템은 화면을 페이지 단위로 나누어 사용할 뿐, 공유한다는 개념이 아니다. 또한 각 화면에서의 문자 입력도 라인 단위로 입력이나 통신 가능하다. 또한 통신 기술도 실질적인 문자 코드를 전달하고 있어 전송 속도에 영향을 주고 있다[15].

IBM 의 Xgroupsketch 시스템은 공유 화면에 영어 문자를 사용하지만 각 문자를 코드로 사용하기 때문에 객체로서의 취급은 불가능하다[14]. Xerox PARC 에서 설계한 전자 회의 시스템인 Colab 는 회의 지원을 위해 Boardnoter, Cognoter, Argnoter 의 3 종류의 툴이 있으나 공유 윈도우 입력은 가능하나 벡타 제어는 설계되지 않았다[10]. 일본 전기(주) C&C 시스템 연구

소의 멀티미디어 분산 회의 시스템 MERMAID (Multimedia Environment for Remote Multiple-Attendee Interative Decision-Making) 는 탁상 회의 시스템으로, 멀티미디어 문서의 검색 편집이 가능하며, 공유 윈도우의 엑세스 권한을 가진다[4]. Bellcore 의 Quilt 는 공유 문서에 주석 달기, 집필자 간의 전자 메일로 메시지 교환 등의 기능을 제공하는 비실시간 그룹 문서 작성 시스템이다[12].

이러한 시스템들은 모두 멀티미디어 전송이나 문서 작성용이다. 이러한 시스템을 이루기 위한 기반 기술로는 주로 비트맵 편집기를 사용하는 인터페이스를 사용하고 있다[12,13,14]. 그러므로 각 객체에 대한 시스템 내부의 접근이 어렵다.

또한 객체를 공동작업시 공유하게되면 같은 객체를 사용하는데 사용자에 따른 일관성 유지가 매우 큰 문제이다[1,2,5,9]. 데이타베이스 시스템에서는 자료를 공유함에 있어서 타사용자가 엑세스하지 못하도록 하는 동시성 제어를 사용한다[5,9].

그러나 공동작업에 의한 객체의 제어는 자료의 제어보다는 공유화면에 나타나는 객체를 제어하는게 목적이다.그러므로 공동작업에서의 객체의 일관성 유지는 동시성 제어보다는 일관성을 유지하려 노력하는데 이러한 방법은 여러 방법이 있다[1,5,9,10].

SUITE 에서의 동시성 제어는 공동 작업시의 권한을 가지고 지료를 관리한다. 이때 각 사용자에게다차원이고 상속에 근거를 둔 방법을 사용한다[5]. 이 때, 읽기, 쓰기, 보기등의 권한을 가진다. 또한 상속에 관한 방법은 보호된 객체나 공동 엑세스 제어권을 나타낸다 [5]. 공동작업 시스템인 GROVE[7] 는 복제 방식을 사용하는 편집기이다[7]. 이 GROVE 는 여러 뷰를 제공하는데, 이 뷰는 공동 윈도우에 나타나다. 이 윈도우는 private, shared, public 이다[7]. 이 GROVE 에서의 동시성 제어는 operation transformation 을 사용한다[7]. 이 방법은 롤백은 예방할 수 있지만, 오퍼레이션의 정확한 순서를 제시하지는 못한다. 그러므로 이방법은 융통성을 제공하고 있지 못하다. 이러한 방법은 공동작업시 지우거나 고치는등의 객체에 대한 접근이 빈번한 한글 에이전트로서는 적합하지 않다

본 논문에서는 중앙 집중식과 복제식 인터페이스 구조의 장점을 가진 새로운 혼합식 한글 에이전트의 다중사용자 인터페이스의 구조를 제시하고 이 다중사용자 인터페이스에서 객체의 일관성 제어를 위한 알고리즘을 제시한다.

이 인터페이스에서는 응용 프로그램이 각 사용자에게 복제되어지고, 각 응용 프로그램이 각 사이트에서 자료를 공유 자료 객체로서 가지게된다. 그러나 객체의 일관성을 유지하기 위한 알고리즘은 중앙 집중적 코디네이터에서 실행되어진다. 본 한글 에이전트는 CSCW 를 위한 다중 사용자용 객체지향 인터페이스를 제공하는 에이전트로서, 정보의 전달은 실제의 객체나 자료가 아닌 이벤트에 의하여 전달, 처리된다. 한글 정보는 객체지향 기술로 처리되며, 오퍼레이션의 의미에 의하여 공유 객체의 일관성을 유지하고 있다.

본 한글 에이전트의 특징은 모든 한글 정보를 객체 단위로 처리하는데,+ 이는 객체지향 인터페이스가 제공하는 여러 잇점을 가진다. 이 다중사용자 인터페이스는 오퍼레이션의 신속한 되돌림을 제공하게되며, 일관성 유지를 통하여객체의 첨가와 삭제를 쉽게하여 준다.

제 2 절에서는 전반적인 한글 에이전트와 다중사용자 인터페이스 모델에 대하여 설명하고, 제 3 절에서는 다중사용자 인터페이스에서 객체의 일관성 유지 방법을 설명한다. 제 4 절에서는 구현 결과와 성능을 평가한다. 제 5 절에는 결론이 있다.

2. 한글 에이전트

본 절에서는 한글에이전트의 개발 모델, 동작 모델, 제어 모델, 다중 사용자 인터페이스 모델을 설명한다.

2.1 한글 에이전트 모델

본 논문에서 제시하는 한글 에이전트 개발의 접근 방식은 크게 외부 즉, 한글 에이전트의 응용 모델과 소프트웨어 구조로 구별되어진다. 응용 모델은 한글의 태스크 모델, 메타포와 사용자 인터페이스 레벨의 객체 모델, 소프트웨어 레벨의 객체 모델로 구성된다. 소프트웨어 구조는 시각적 사용자 인터페이스 서브시스템과 응용 기능의 핵심 서브 시스템으로 구성된다. 시각적 사용자 인터페이스 서브시스템은 객체 중심 구조적 라이브러리와 문맥적인 기능으로 구성된다. 그림 2-1 은 한글 에이전트의 태스크 모델과 객체 모델의 관계를 나타낸다.

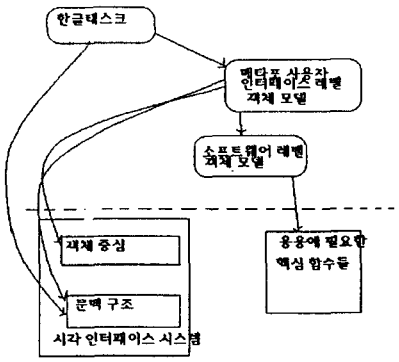

그림 2-1 한글 에이전트의 태스크 모델과 객체 모델의 관계

2.2 한글 에이전트의 동작 모델

본 절에서는 객체지향 한글 에이전트의 동작 모델을 설명한다. 한글 에이전트의 동작 모델은 한글공동작업의 시작, 새로운 참석자의 참석, 한글공동작업의 실행, 한글공동작업시 객체에 대한 다중접근의 제어, 한글공동작업의 종료의 동작 모델을 가지는데 동작 모델은 그림 2-2 와 같다. 그림 2-2 에서 사이트 프로세스와 코디네이터가 각각의 고유 동작을 실행하고 있다.

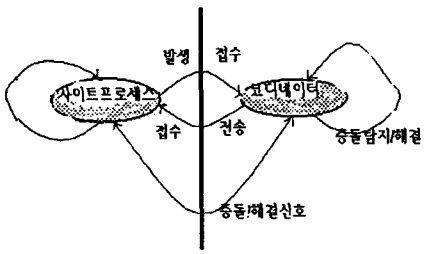

그림 2-2 한글 에이전트의 동작 모델

한글공동작업의 시작 모델은 한글공동작업 소집자에 의하여 이루어 지는데, 소집자는 한글공동작업 관리기를 수행하고 자신의 한글 에이전트를 수행시킨다. 이때, 한글공동작업 관리기는 한글공동작업의 시작을 알린다. 그후 한글공동작업 통지를 받은 사용자는 한글 에이전트를 수행하여 한글공동작업 관리기와 연계를 이룬다. 이렇게 하여 사용자의 한글 에이전트는 한글공동작업 관리기와 연계를 이루고 한글공동작업 관리기를 통해 다른 사용자의 한글 에이전트와 연계를 이룬다.

새로운 사용자의 참석은 사용자가 자신의 한글 에이전트를 실행시킬 때 이루어진다. 새로운 사용자가 자신의 한글 에이전트를 수행하면 한글공동작업 관리기와 연계가 이루어지고 한글공동작업 관리기로 부터 한글공동작업 정보 테이블의 내용을 받아 한글공동작업에 참석하게 된다.

한글공동작업의 실행 모델은 한글공동작업 참석자가 공동작업을 수행할 때 이루어진다. 참석자의 편집행위는 한글공동작업 관리기를 통해 다른 참석자의 한글 에이전트로 전달되며 한글공동작업 관리기에서 객체에 대한 다중다중 접근을 감지하고 제어한다.

한글공동작업의 종료모델은 한글공동작업의 소집자에 의하여 이루어지는데, 소집자는 모든 참석자들에게 한글공동작업의 종료를 알린 후 한글공동작업의 내용을 저장한다.

이와같은 일을 위하여 각 사이트 프로세스와 코디네이터는 다음과 같은 과정을 지닌다.
사이트 프로세스의 기능은 다음과 같다.

1) *오퍼레이션 발생*: 오퍼레이션은 각 사이트의 사용자에 의하여 생성되어진다. 사이트 프로세스는 코디네이터 프로세스로 보내어질 오퍼레이션 메시지 내에 있는 오퍼레이션을 캡슐화한다.
2) *오퍼레이션 접수*: 여러 사이트에서 생성되어진 오퍼레이션 메시지는 코디네이터를 통하여 접수되어진다.
3) *오퍼레이션 수행*: 접수되어진 오퍼레이션이 실행되어진다.
4) *충돌 탐지*: 객체 사이에서의 일관성이 유지되지 않는 오퍼레이션 메시지를 탐지한다.

코디네이터는 코데네이터 프로세스와 공유 자료 객체로 구성되어지는데, 공유 자료 객체는 사이트 자료 객체와 같다. 코디네이터 프로세스의 기능은 다음과 같다.

1) *오퍼레이션 접수*: 오퍼레이션 메시지가 여러 사이트에서 들어오게된다.
2) *오퍼레이션 전송*: 사이트로 부터 온 오퍼레이션 메시지를 전송한다.
3) *충돌 탐지*: 객체 사이에서의 일관성이 유지되지 않는 오퍼레이션 메시지를 탐지한다.
4) *충돌 해결*: 충돌이 일어나면 코디네이터 프로세스나 어떤 사이트에서 충돌을 해결한다.

2.3 이벤트 드리븐 모델

한글 에이전트의 공유화면 제어, 메시지 전달, 사용자

와의 상호작용은 이벤트 드리븐으로 이루어 진다. 이 이밴트 드리븐의 제어 구조에서 제어 컴포넌트는 사용자의 입력 장치 이벤트, 윈도우 환경 이밴트, 기타 반응을 주어야 하는 기능에 대한 다른 시스템 이벤트를 dispatch 한다. 이러한 모델 컴포넌트는 제어 컴포네트와 더불어 반응한다. 그림 2-3 은 한글 공동작업을 위한 이벤트 드리븐 구조를 나타내고 있다.

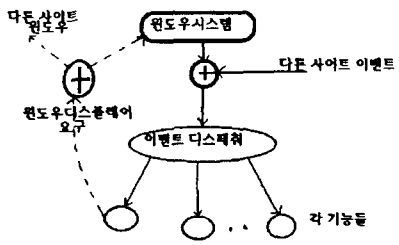

그림 2-3 한글 공동작업을 위한 이벤트-드리븐 구조

2.4 다중사용자 인터패이스 모델

본 한글 에이전트는 인터패이스와 인터패이스 사이의 상호작용에 의하여 동작된다. 다중사용자 인터패이스는 공동작업을 위하여 사용되어지도록 설계, 개발된다[1,14]. 본 한글 에이전트용 다중사용자 인터패이스는 중앙 집중식과 복제식 인터패이스의 특징을 지닌 혼합식 인터패이스 구조이다. 공유 자료와 응용 프로그램은 분산된 각 사이트에 복제되어지고, 이벤트로서 통신과 일관성 유지는 코디네이터라는 서버에 의하여 처리된다[1]. 본 다중사용자 인터패이스는 객체 중심 인터패이스로서 여러 객체를 사용하므로, 각 사용자들은 분산된 환경에서 상대적 위치를 고려하지 않고 독립적으로 자신의 작업을 수행한다. 예를들면 한 사이트의 공유 인터패이스에서

"작업 합시다"

라고 쓰고, 다른 임의의 사이트에서
"시작합니다"
라고 쓰면,
"작업 "시작합니다"합시다"
와 같이 중복되어 쓰여질 수 있다. 그림 2-4 은 공동작업 중 공유 자료인 한글 문자 객체가 공유 화면에 표현되어진 것이다.

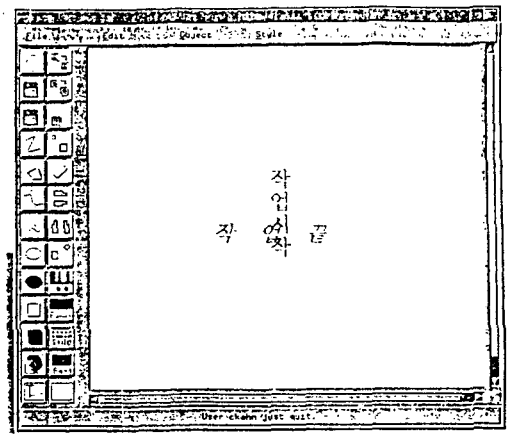

그림 2-4 중복 표현되어진 공유 화면

다중사용자 인터패이스는 통신 기반으로 연결되어진 사용자 인터패이스의 집합으로 이루어진다. 본 논문에서 제시하는 다중사용자 인터패이스는 공동 작업에 각 참가자가 있는 '사이트'에 공유된 자료를 코디네이터라는 조절자를 통하여 표현하기도하고 자료를 저장하기도 한다. 이 코디네이터는 공유 자료를 각 참가자가 있는 사이트로 보내어 인터패이스에 표현하기도하고, 공유된 자료의 변경 사항을 전달하기도하고, 공유 자료의 엑세스를 조절하기도 한다.

다중사용자 인터패이스는 $I = <S, O, C>$ 와 같이 모델화할 수 있는데, 여기서, S 는 사이트들의 집합, O 는 인자가 있는 오퍼레이터들의 집합, C 는 코디네이터이다. 각 사이트는 사이트 프로세스와 사이트 자료 객체로 구성되어 있으며, 각 사이트 자료는 공유되어지는 자료 객체 이다. 다중사용자 인터패이스에서 각 사이트 자료 객체는 사이트 객체 사이에서의 일관성을 유지하도록 하여야 한다. 사이트 자료 객체의 구조는 사용하거나 적용 가능하게 하여야 하는데, 본 한글 에이전트의 한글공동작업기는 한글 객체들을 링크로 연결하고 있다.

집합 O 는 모든 사이트에서 사이트 자료 객체에 정의되어지는 오퍼레이터들로 구성되어지는데, 다중사용자 인터패이스의 한글공동작업기는 각 사이트 자료 객체로서 한글 객체를 가지며, 이것은 $O = \{O_1, O_2, ..., O_n\}$ 와 같이 정의 한다. 때 O_i ,i = 1,.., n, 는 공동작업 중에 변할 수 있다. 예를들면, 오퍼레이션 O_i,i = 1,.., n, 는 다음과 같이 정의된다.

O_i: {자료 객체} × {사이트 자료 객체의 구조} -> {사이트 자료 객체의 구조}

O_i(obj,obj_mng) = create(obj,obj_mng) 로서, obj ∈ { 자료 객체 } 이며, obj_mng ∈ { 사이트 자료 객체의 구조 } 인데, 한글 객체 g 를 생성하고, 사

이트 자료 객체의 구조 obj_mng 에 추가 한다.
오퍼레이션은 오퍼레이터들의 인스턴스이며 각 사이트 자료 객체의 구조를 변경시킬 수 있다. 를들면, 위의 한글공동작업기인 경우에 오퍼레이션 $o(o \in O)$ 는 o = create(letter1_obj, letter2_obj]) 를 구성한다. 그 후 이 오퍼레이션이 끝나면 사이트 자료 객체의 구조는 [letter1_obj, letter2_obj, letter3_obj] 와 같이 변경되어 진다.

한글 에이전트의 다중사용자 인터페이스에서 나타나는 충돌을 해결하는 방법을 증명하기 위하여 몇 개의 정의를 할 수 있다.

정의 1: 사이트 s 에서 발생되어진 오퍼레이션 o_1 와 o_2 가 있을 때, o_1 가 o_2 전에 발생되었으며, 사이트 s 에 서 o_1 가 o_2 전에 실행되어지면 partial ordering 을 만족한다고 한다.

정의 2: s_1 와 s_2 에서 발생되어진 오퍼레이션 o_1 와 o_2 가 있을 때, o_1 가 o_2 보다 먼저 발생되고, 임의 의 사이트인 s_1 와 s_2 에서 o_1 가 o_2 보다 먼저 실행되면, total ordering 을 만족한다고 한다.

정의 3: 사이트 s 에서 생성된 오퍼레이션 o 가 주어졌을 때, 오퍼레이션이 total ordering 을 만족하면 오퍼레이션이 타당하다고 하고 그렇지 않으면 부당하다고 한다.

정의 4: 코디네이터나 어떤 사이트에서 부당한 메시지가 발견되어질 때, 다중사용자 인터페이스는 충돌 상태에 있다고 한다.

정의 5: 사이트에서 발생되어진 오퍼레이션들이 없고, 실행을 기다리거나 transit 상태에 있는 오퍼레이션들이 없으면 비활동 중에 있다고 한다.

정의 6: 모든 사이트 객체가 비활동 중에 객체 동치성을 가지면, 사이트 자료 객체는 일치성이 있다고 한다.

정의 7: 사이트 자료 객체가 일치성을 가지면, 다중사용자 인터페이스는 정확하다고 한다.

위의 정의에서 보듯이, 오퍼레이션의 total ordering 은 객체가 일치성을 가지기 위한 충분조건 처럼 보이지만 어떤 오퍼레이션에 대하여는 그렇지 않다. 예를들어 다중사용자시스템이 $I = \langle \{s_1, s_2\}, \{delete\}, C \rangle$ 라 하자. 오퍼레이션 o_1 = delete(obj_1) 가 사이트 s_1 와 s_2 에서 동시에 발생되었다고 한다면, 한 오퍼레이션은 의미적으로 중복되어진다. 이는 total ordering 이 오퍼레이션의 의미에서 다중사용자 인터페이스 객체의 일치성을 유지하기 위하여 항상 만족되어야 하는 것은 아니라는 것을 의미한다.

이러한 모델에서 발생되어진 오퍼레이션은 우선 사이트에서 실행되어지고 난 후, 사이트 자료 객체의 자료와 사이트 자료 객체의 구조를 변경한다. 실행이 끝난 후에는 오퍼레이션 메시지가 코디네이터로 전달되어지는데, 코디네이터 프로세스는 오퍼레이션의 타당성을 검사한다. 만약 오퍼레이션 메시지가 타당이면 오퍼레이션은 코디네이터 프로세스에 의하여 실행되어진다. 이는 코디네이터 내에 있는 자료 공유 객체의 스테이트 변화시킨다. 그 후, 오퍼레이션 메시지를 보냈던 사이트는 물론 모든 사이트에 그 상태를 전달한다.

만약 오퍼레이션 메시지가 부당하다면, 중앙 코디네이터 프로세스는 부당한 오퍼레이션을 생성하게된 충돌의 타입을 검사하고, 이 충돌의 타입에 따라서 그 충돌을 해결한다.

한글 에이전트 시스템이 정확하게 하려면 두 개의 문제가 있는데, 하나는 부당한 오퍼레이션을 어떻게 탐지할 것인가? 이고, 하나는 부당한 오퍼레이션이 발견되었더라도 어떻게 그 상황을 해결할 것인가? 이다.

여기서 순서-종속 (Sequence-Dependent) 오퍼레이션 카운터와 지역 객체 참조 카운터를 이용하여 부당한 오퍼레이션을 탐지하는 방법과 오퍼레이션-의미 (*Semantics of Operations*) 를 이용하여 부당한 오퍼레이션을 해결하는 방법을 제시한다.

3. 공유화면 객체의 일관성 유지

본 절에서는 오퍼레이션의 의미에 근거를 둔 다중사용자 인터페이스를 가지는 한글 에이전트의 일관성을 유지하는 알고리즘을 제시한다. 이 알고리즘은 다중사용자 인터페이스에게 알맞은 여러 특성을 제시하게된다. 먼저, 오퍼레이션은 발생되어진 사이트에서 즉시 실행되어지고, 즉시 신속한 피드백이 이루어 진다. 그 다음으로는 이 알고리즘이 여러 사이트에 있는 다중사용자 인터페이스에게 각 상태를 전달한다.

충돌을 탐지하는 알고리즘은 순서-종속 (Sequence-Dependent) 오퍼레이션 카운터와 지역 객체 참조 카운터를 사용한다. 순서-종속 오퍼레이션 카운터는 SDO-SDO (Sequence-Dependent Operation)의 충돌을 탐지하는데 사용되고, 지역 객체 참조 카운터는 SIO-SIO (Sequence-Independent Operation)를 탐지하는데 이용되어진다.

본 절에서 충돌을 탐지하는 알고리즘을 위하여 오퍼레이션의 의미를 제시하는데, 이 방법을 설명하기 전에 오퍼레이션의 의미와 오퍼레이션의 의미를 바탕으로하는 충돌의 타입을 설명한 후, 알고리즘 대한 자료 구조를 설명한다.

3.1 오퍼레이션의 의미

한글 에이전트가 효과적으로 공동 한글 작업을 수행되도록 하기 위하여는 여러 오퍼레이션의 의미가 있게 된다. 여러 개의 다른 객체를 생성하는 것과 같은 오퍼레이션들은 사이트 자료 객체 구조를 변화 시키기 때문에 모든 사이트에서 같은 순서로 실행되어져야만 한다. 아니면, 사이트 자료 객체 구조는 서로 다르거나, 오퍼레이션이 실행되고 나서는 일관성이 없게된다. 그러므로 이는 시스템을 정확히 작동되지 않도록 한다. 다른 객체를 삭제하는 어떤 오퍼레이션은 사이트 자료 객체 구조를 변화시키지만, 다른 순서로 오퍼레이션이 실행된 후에 사이트 객체와 객체 구조의 상태가 같거나 일치성을 가지기 때문에 같은 순서로 실행되어질 필요는 없다.

여기서, 오퍼레이션의 의미에 따라 오퍼레이션을 분류하면 다음과 같다.

순서-종속 오퍼레이션(*Sequence-Dependent Operation:SDO):* 이 오퍼레이션은 모든 사이트에서 같은 순서로 실행되어야만 한다. 즉, 이 오퍼레이션은 total ordering 의 특성을 만족하여야만 한다. 객체를 생성하는 오퍼레이션인 creation 은 이 오퍼레이션 타입에 들어간다.

순서-독립 오퍼레이션(*Sequence-Independent Operation (SIO):* 이 오퍼레이션은 같은 순서로 실행되어질 필요는 없다. 즉, 이 부류에 속하는 오퍼레이션은 total ordering 특성을 만족하지 않는다. 이 경우에 total ordering 에 따라 실행되어지는 오퍼레이션은 그 인자로 같은 객체를 가지는 오퍼레이션인 경우에 한글 에이전트를 정확히 작동되지 않게 할 수도 있다. 이러한 오퍼레이션으로는 객체의 삭제, 변경을 위한 deletion, updating 등이다.

3.2 충돌의 타입

여기에서는 오퍼레이션의 의미에 바탕을 둔 충돌의 타입을 정의 한다. 오퍼레이션을 순서-종속 오퍼레이션 (Sequence-Dependent Operations:SDO) 과 순서-독립 오퍼레이션 (Sequence-Independent Operations:SIO) 으로 분류하는데, 세 종류의 충돌 타입이 있다.

SDO-SDO 충돌 : 이 오퍼레이션은 순서-종속 오퍼레이션(Sequence-Dependent Operations:SDO) 으로서 순서-종속 오퍼레이션이 만드는 충돌이다. 즉, 순서-종속 오퍼레이션은 total ordering 의 특성을 만족하지 않는다. 이미 알듯이 순서-종속 오퍼레이션은 충돌을 해결하기 위하여 오퍼레이션이 total ordering 이 되도록 순서 지어져야 한다.

SIO-SIO 충돌 : 이 오퍼레이션은 순서-독립 오퍼레이션(Sequence-Independent Operations:SDO) 으로서 순서-독립 오퍼레이션이 만드는 오퍼레이션이 만드는 충돌이다. 즉, 순서-독립 오퍼레이션은 오퍼레이션의 total ordering 이 곧 시스템을 정확히 작동되지 않게 한다,

SDO-SIO 충돌 : 이 오퍼레이션은 순서-종속 오퍼레이션(Sequence-Dependent Operations:SDO) 과 순서-독립 오퍼레이션(Sequence-Independent Operations:SDO) 이 만드는 충돌이다.

충돌 해결은 충돌의 타입에 따라 다르게 적용되어 지는데, 한 예로, 다중사용자 인터페이스가 $I = < \{s_1, s_2\}, \{create, update, delete\}, C >$, 이고, create 오퍼레이션은 순서-종속 오퍼레이션이며, update 나 delete 오퍼레이션은 순서-독립 오퍼레이션이다. 이 때, 오퍼레이션 o_1 = create(obj_1) 이 사이트 s_1 에서, 오퍼레이션 o_2 = create(obj_2) 이 사이트 s_2 에서 동시에 발생되어지면, 그림 3-1 와 같은 SDO-SDO 충돌이 나타난다.

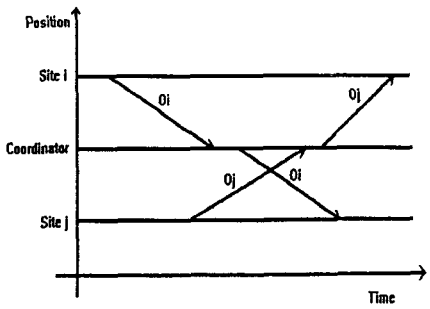

그림 3-1 SDO-SDO 충돌

이 경우의 충돌은 오퍼레이션의 total ordering 에 의해 해결되어진다. 오퍼레이션의 total ordering 은 오퍼레이션 우선 순위와 순서-종속 오퍼레이션 카운터의 값에 결정되어진다. 오퍼레이션의 otal ordering 을 얻기 위하여 시스템은 오퍼레이션 롤백을 수행하고, 오퍼레이션의 total ordering 을 결정한다.

또 다른 경우를 살펴보면, 사이트 s_1 에서 오퍼레이션 o_1 = delete(obj_1) 이, 사이트 s_2 에서 오퍼레이션 o_2 = delete(obj_2) 이 동시에 발생하면, SIO-SIO 충돌이 발생한다. 이 경우에 충돌 해결 방법은 오퍼레이터와 인자 객체에 따라 세 개의 다른 방법이 있다.

먼저, 객체 obj_1 가 객체 obj_2 와 같은 객체이고, 두 오퍼레이터가 같으면, 한 오퍼레이션은 중복된다.그러므로 충돌을 해결하는 방법은 한 오퍼레이션을 무시하는 것이다. 둘째, 객체 obj_1 가 객체 obj_2 와 다르다면, 오퍼레이션이 실행된 후에도 사이트 자료 객체의 일관성이 유지되기 때문에 충돌을 해결할 필요가 없다. 또한 오퍼레이션 o_2 = delete(obj_2) 가 아닌 o_2 = update(obj_2) 가 사이트 s_2 에서 발생하고, 객체 obj_2 가 객체 obj_1 와 다르다면, 충돌 해결은 불필요 하다. 그러나 객체 obj_2 가 객체 obj_1 와 같으면, 오퍼레이션의 total ordering 이 충돌을 해결하는데 부적당하다. 오퍼레이션의 total ordering 이 [delete(obj_1), update(obj_2)] 이 되는 경우가 있기 때문이다. 이 경우에, 오퍼레이션 한 개는 무시되어진다. 무시되어질 오퍼레이션을 결정하는 것은 오퍼레이션 우선 순위에 기반을 두고 있다.

3.3 자료 구조

앞에서의 알고리즘이 실행되면서 사용하게될 자료 구조를 설명한다.

순서-종속 오퍼레이션 카운터(Sequence-Dependent Operation Counter:SOD-Counter): 이 카운터는 순서-종속 오퍼레이션 카운터로서 코디네이터와 모든 사이트에 자신의 카운터를 가지고 있다. 사이트 프로세스에서는, 사이트에서 발생되어지거나 코디네이터에서 전달되어진 의미-종속 오퍼레이션이 실행되어지면, 의미-종속 오퍼레이션 카운터가 증가된다. 코디네이터에서는 의미-종속 오퍼레이션이 전송되면, 순서-종속 오퍼레이션 카운터가 증가되어진다.

지역 객체 참조 카운터(Local Object Reference Counter:LO-Counter): 이 카운터는 객체를 참조하는 카운터이다. 공유 자료 객체 속에 있는 각 객체는 지역 객체 참조 카운터를 가지게되는데, 오퍼레이션이 객체를 엑세스 할 때 마다 증가되어진다. 즉, 오퍼레이션이 그 오퍼레이션의 인자로서 객체를 가지게 된다.

오퍼레이션 메시지(Operation Messages:OM): 이 메시지는 <*i, o, p, v, w*> 의 형태를 가지는데, *i* 사이트 식별자, *o* 는 오퍼레이션, *p* 는 오퍼레이션 *o* 의 우선순위,*v* 는 *o* 오퍼레이션 속에 있는 인자의 지역 객체 참조 카운터의 값이고, *w* 는 SOD-카운터의 값이다.

오퍼레이션 메시지 큐(Operation Message Queue:OMQ): 사이트 오퍼레이션 메시지큐는 사이트에서 실행되기 위해 기다리고 있는 오퍼레이션 메시지를 기록하고 있다. 오퍼레이션 메시지가 코디네이터 프로세스로 부터 전달되어지면 그 내용이 추가되어진다. 코디네이터 메시지 큐는 각 사이트로 전송 되어지기 위해 기다리는 오퍼레이션들을 대기 시킨다.

오퍼레이션 메시지 로그(Operation Message Log:OML): 각 사이트 프로세스는 각 사이트에서 실행되어지는 오퍼레이션 메시지의 로그를 유지한다. 이 로그는 삽입에 의하여 순서화되어진다.

3.4 충돌 탐지와 해결 알고리즘

본 절에서는 오퍼레이션의 의미 에 의한 충돌의 탐지 방법과 해결 방법을 설명한다. 충돌은 코디네이터 프로세스와 모든 사이트 프로세스에서 탐색되어진다. 여기에서 몇개의 가정을 필요로 한다. 즉, 같은 인자 객체로 순서-종속 오퍼레이션으로 충돌이 생기는 순서-독립 오퍼레이션이 없다고 가정한다. 이러한 가정을 가짐으로서 알고리즘은 좀 융통성이 있게될 것이다. 다중 사용자 인터페이스에서는 객체를 생성하는 오퍼레이션은 순서-종속 오퍼레이션이고, SDO-SIO 충돌은 결코 일어나지 않기 때문이다.

이런 가정을 통해서 SDO-SDO 가 문제가 되는데, 이는 같은 오퍼레이터를 가지고 다른 인자 객체를 가지는 오퍼레이터에 의해 생기는 충돌이기 때문이다. SIO-SIO 충돌은 인자로서 같은 객체를 가지는 오퍼레이터에 의해 생기는 충돌이다.

다음은 코디네이터 프로세스에 의하여 실행되어지는 과정이다.

초기화

코디네이터 프로세스는 순서-종속 오퍼레이션 카운터와 오퍼레이션 메시지 로그를 초기화 한다.

접수 오퍼레이션

여기에서는 어떤 사이트에서 전달되어진 오퍼레이션 메시지를 받아들이고, 앞으로 전송되어지기 위하여 오퍼레이션 메시지 큐에 넣는다. 이후 오퍼레이션 메시지 큐에 있는 오퍼레이션 메시지에 의하여 야기 되어질 수 있는 충돌을 탐지한다.

충돌 탐지

충돌의 탐지는 오퍼레이션 메시지 큐에 있는 오퍼레이션 메시지를 하나 씩 꺼내어서 충돌을 탐지한다. 이때 각 오퍼레이션의 순서에 대하여 오퍼레이션의 의미가 순서-종속이고, 코디네이터의 SDO-카운터 값이 코디네이터에 의한 오퍼레이션 메시지의 값 보다 작지 않으면, 이 오퍼레이션 메시지는 SDO-SDO 충돌이 일어난다. 오퍼레이션의 의미가 순서-독립이고, 오퍼레이션 메시지 내에 있는지역 카운터 값이 공유 자료 객체 속에 있는 객체의 지역 카운터 값 보다 크지 않으면 SIO-SIO 충돌이 일어난다. 또한 충돌이 발생 되더라도, 충돌 해결이 반드시 필요한 것은 아니다. 충돌이 탐지되면 충돌은 해결 모듈에 의하여 해결되어지는데, 이 모듈은 오퍼레이션 메시지 로그를 가지며, 충돌을 야기시키는 부정확한 오퍼레이션 메시지를 처리하여 충돌을 해결한다. 충돌이 일어나지 않으면 오퍼레이션은 모든 사이트에 전송되어진다.

충돌 해결

SDO-SDO 충돌이 탐색되어지면, 오퍼레이션의 total ordering 이 필요하게된다. 가장 최근의 오퍼레이션으로 부터 오퍼레이션 메시지 로그를 검사하면, 부정확한 오퍼레이션 메시지에 의한 충돌이 발생하는 순서-종속 오퍼레이션을 발견하게된다. 그러면 오퍼레이션의 순서에 근거하여 이 오퍼레이션들을 순서대로 정리한다. 오퍼레이션의 total ordering 후에 사이트에 종속되어진적당한 오퍼레이션 롤백 요구를 가지는 순서가 결정되어진 오퍼레이션 메시지를 전송한다. SIO-SIO 충돌이 발생되면, 가장 최근의 오퍼레이션에서 출발하여 오퍼레이션 메세지 로그를 검사하여 부정확한 오퍼레이션으로 충돌을 야기시키는 순서-독립 오퍼레이션을 찾아내다. 만약 오퍼레이션이 부정확한 오퍼레이션과 같으면, 부정확한 오퍼레이션을 무시하고, 오퍼레이션이 인자 객체를 제외한 부정확한 오퍼레이션과 다르면, 그 오퍼레이션은 우선 순위를 비교하게된다. 이 경우에 부정확한 오퍼레이션이 우선 순위가 낮으면, 오퍼레이션을 무시하고 부정확한 오퍼레이션이 실행되지 않도록 적당한 롤백 요구를 사이트에 전달한다. 만약 오퍼레이션의 우선 순위가 더 낮으면, 부정확한 메시지를 보내온 사이트를 제외한 모든 사이트에 적당한 롤백 요구를 전달한다.

다음은 사이트 프로세스에서 실행되어지는 과정이다.

초기화

각 사이트 프로세스는 순서-종속 오퍼레이션 카운터, 오퍼레이션 메시지 큐, 오퍼레이션 메시지 로그를 초기화한다.

접수 오퍼레이션

이 오퍼레이션은 코디네이터로 부터 오퍼레이션 메시지를 받아들이고, 앞으로 실행되기 위하여 오퍼레이션 메시지 큐에 넣는다.

오퍼레이션의 발생

사용자 인터페이스로 부터 오퍼레이션을 받아들이고, 오퍼레이션 메시지를 만든다음 이 오퍼레이션 메시지를 코디네이터로 보내어 실행시킨다.

오퍼레이션 실행

사용자에 의하여 오퍼레이션이 발생되어지면 즉시 그 오퍼레이션은 실행되어진다. 이 오퍼레이션이 실행되고 나서는 오퍼레이션 메시지 큐를 검사하게된다. 만약 오퍼레이션 메시지 큐에 이 오퍼레이션이 있으면, 오퍼레이션의 부정확성을 검사한다. 즉, 충돌 탐색 모듈을 검사한다. 이때 충돌이 탐색되어지면 코디네이터에 부정확한 오퍼레이션과 함께 충돌 해결 요구를 보내게 된다. 충돌이 없으면 오퍼레이션은 실행되어진다.

충돌 탐색

충돌 탐색 모듈은, 부정확한 오퍼레이션의 의미가 순서-독립이고, 이 오퍼레이션이 오퍼레이션 메시지 로그 에 있을 때는 (실행하고 있다면) 그 오퍼레

이션이 무시되어진다는 것을 제외하곤, 코디네이터 프로세스와 거의 같은 방법으로 실행되어 진다.

4. 구현과 성능 평가

본 공동작업을 위한 한글 에이전트 시스템은 UNIX의 X-Window 환경에서 OSF/Motif 와 C++ 를 사용하여 설계되었다. 여러 사이트의 워크스테이션을 통하여 한글의 공동작업을 위한 객체 지향 편집이 가능하다. 그림 2-4 와 같은 다중사용자 인터페이스에서 왼쪽의 두 열은 각 기능을 나타내는 아이콘을 표현하였고, 오른쪽은 공유 가능한 한글공동작업 공간을 나타낸다. 기타 여러 기능이 이 있다[1]. 각 아이콘 들은 한글공동작업을 위한 제어 아이콘, 자원 관리를 위한 아이콘, 작업을 위한 아이콘, 화면 제어를 위한 아이콘과 작업을 위한 아이콘이 있다[1]. 각 아이콘을 사용하면 공동작업에 필요한 화면이나 윈도우가 나타나고 참가자 사이에서의 공동작업의 연계가 나타난다.

이 한글 에이전트 시스템에서는 공유 인터페이스에 나타나는 객체의 일관성을 유지 한다. 그러므로, 충돌을 탐지하고, 순서-종속과 순서-독립 오퍼레이션으로 구별되는 오퍼레이션의 의미에 근거를 두고 해결 한다.

SDO-SDO 충돌의 경우에는 충돌탐지 알고리즘이 간단하고 빠르다. 이러한 타입의 충들이 거의 없으면, 이 한글 에이전트 시스템의 성능은 아주 좋게된다. 충돌이 일어날 때, 충돌 해결을 위하여 오퍼레이션의 total ordering 이 필요하게 된다. 이것은 곧 충돌 해결이 오퍼레이션의 롤백을 요구하고, 시스템이 정상적이지 않다는 것을 의미한다. 이렇게되면, 다중사용자 시스템에서는 오버헤드가 매우 커진다.

SIO-SIO 충돌의 경우에는 오퍼레이션의 total ordering 이 불필요하게되며, 충돌을 야기하는 두 오퍼레이션이 같으면, 충돌을 해결하기 위하여 한 오퍼레이션을 무시하게된다. 이 경우에는 시스템에 오버헤드가 없다는 것을 의미한다. 만약 두 오퍼레이션이 다르면 시스템은 낮은 순위의 한 오퍼레이션을 무시하고, 이 오퍼레이션을 발생시킨 사이트로 오퍼레이션을 롤백한다. 이경우는 SDO-SDO 충돌과 같은 시스템 오버헤드를 가진다.

본 다중사용자 인터페이스 모델에서의 시스템 성능은 충돌의 수에 반비례한다.

5. 결 론

본 한글 에이전트 시스템은 다중사용자 인터페이스와 통신 부분으로 나누어진다. 다중사용자 인터페이스에서는 그래픽스, 텍스트, 이미지 등과 한글을 처리하는 한글 에이전트를 제공하며 한글공동작업시 한글 자료들은 객체 단위로 취급한다. 한글공동작업을 지원하는 그룹 통신은 다대다 통신이 가능하다.

본 논문에서의 다중사용자 인터페이스는 중앙집중식과 복제 방식의 구조를 혼합한 새로운 다중사용자 인터페이스 모델을 제시하였고, 이 인터페이스 모델에서의 객체의 일관성 유지 알고리즘을 제시하였다. 이 인터페이스 모델에서는 응용 프로그램의 복제는 각 사이트의 사용자에게 복제되어 사용되고 각 응용 프로그램은 각 사이트의 자료로서 공유 자료 객체를 가진다. 그러나 일관성 제어는 코디네이터에서 수행되어진다.

본 논문에서는 오퍼레이션의 의미 의 방법을 제시하였고, 이를 구현하기위하여 오퍼레이션의 의미 의 종류를 분류하였다. 이러한 분류는

1) 모든 사이트에서 같은 순서로 실행되어져야만 하는 오퍼레이션들,

2) 실행 순서와는 무관한 오퍼레이션들

로 구분되어진다.

이러한 의미 타입에 따라, 각기 다른 알고리즘이 객체의 일관성을 유지하는데 적용되어진다. 이러한 모델에서는 오퍼레이션을 신속히게 되돌려 줌으로써 공동작업을 위한 효과적인 다중사용자 인터페이스를 지원한다.

앞으로의 연구는 공동작업 중에 공유 객체나 공유 화면의 개인화할 수 있도록 하며 각 사이트의 개인 객체를 개인이 보호 할 수 있도록 한다. 이와같이 함으로써 한글 에이전트에 의한 한글공동작업을 지원함으로써 멀티미디어 원격 회의, 가상 강의, 공동 연구등에 적용 되어진다.

참고문헌

[1] 김 상욱, 안 춘근, 진 윤숙, "그룹편집기에서 의 객체에 대한 다중 접근의 제어", HCI '94 학술대회 발표 논문집, 제 3권, 제 1호, pp.875-878, 1994, 2.

[2] 송 동호, 황 진경, 박 헌일, 서 혜경, "멀티미디어 CSCW를 위한 에이젼트 및 그래픽 사용자 인터페이스", HCI '94 학술대회 발표 논문집, 제 3 권, 제 1 호, pp. 155-169, 1994, 2.

[3] 임 재홍, 박 용진, "공동작업을 지원하기 위한 그룹 통신 플랫폼의 설계 및 구현", 정보과학회 논문지, 제 21 권, 제 1 호, 1994.

[4] Watabe, k., Sakata, S., Maeno, K., Fukuoka, H., and Ohmori, T., "Distributed Multiparty Desktop Conferencing System: MERMAID," ACM 1990 Conference on CSCW, pp. 27-38, 1990.

[5] K. Narayanaswamy and Neil Folman, "'Lazy' Consistency: A Basis for Cooperative Software Development," ACM 1992 Conference on CSCW, pp. 27-38, 1990.

[6] Ellis, C.A, Gibbs, S.T, and Rein, G.L, "Groupware: Some Issues and Experiences," Comm. of ACM, Vol. 34, No. 1, pp. 680-689.

[7] Michel Beaudouin-Lafon, and Alain Karsenty, "Transparency and Awareness in a Real-Time Groupware System," Procedings of the ACM Symposium on User Interface Software and Technology, pp. 171-180, November, 1992.

[8] Gary M. Olson and Judith S. Olson, "Defining a Metaphore for Group Work," IEEE Software, Vol. 9, No. 3, pp. 93-95, May, 1992.

[9] HongHai Shen and Prasun Dewan, "Access Control for Collaborative Environments," ACM 1992 Conference on CSCW, pp. 51-58, November, 1992.

[10] Mark SteFik, Gregg Foster, Daniel G, Bobrow, Kenneth Kahn, Stan Lanning and Lucy Suchman, "Beyond the Chalkboard: Computer Support for Collaboration and Problem solving in Meeting," Comm. of ACM, Vol. 30, No. 1, pp. 32-47, January, 1987.

[11] Alok Sinha, "Client-Server computing," Comm. of ACM, Vol. 35, No 7, pp. 77-98, 1992.

[12] Leland, M.D.P. Fish, R.S., and Kraut, R.E., "Collaborative Document Production Using Quilt," ACM 1988 Conference on CSCW, pp. 206-215, 1988.

[13] Hiroshi Ishii and Naomi Miyake, "Toward an Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation," Comm. of ACM, Vol. 34, No. 12, pp. 37-50, December, 1991.

[14] ____, XgroupSketch, University of California, Berkeley, 1993.

[15] Stevens, UNIX Network Programming, Prentice Hall, Inc, 1991.