

WWW을 이용한 에이전트 기반 공동 설계 환경 개발*

안상준*, 이수홍**

Agent-based Collaborative Design Environment using WWW

Sang June Ahn* and Soo Hong Lee**

ABSTRACT

This paper deals with a development of the system that implements a collaborative design environment with some intelligent agents on the WWW. Intelligent agents can design collaboratively through an interchange of messages in their special domains. Such a collaborative design of agents is achieved on the WWW. In this paper, we propose special agents named intercessor and DCM (Dynamic Connection Manager) and suggest new connection architecture using these agents in the WWW in order to improve the pre-existed agent connection architecture. The proposed agents are developed using Java language and JATLite API^[1]. We apply these agents to the new architecture and show some possibilities that the agent connection architecture can be extended in the WWW. Agents interchange messages with others using KOML (Knowledge Query and Manipulation Language)^[2, 3], agent communication protocol and language, and deal with message autonomously according to their rule base. Agents register and connect dynamically through the intercessor agent, and infer from their.

Key words : Agent, WWW, Collaborative Design, Intercessor, DCM, KQML

1. 서 론

컴퓨터의 진보와 더불어 1990년대에 하나의 쟁점으로 등장한 것이 바로 인터넷이다. 전세계에 걸친 고속 통신망을 이용하여 인터넷은 지금까지 해결하지 못했던 공동 협력 분산 작업을 거의 동시에 해결할 수 있는 기반을 마련하였다. 또한 인터넷과 연계된 이러한 접근 방법은 급속하게 변화하는 제품 개발의 추세와 생산비용의 절감, 가용한 서비스의 활용, 신속한 제품 설계라는 취지와 맞물려 국가와 기업, 학계간에 하나의 큰 과제로 부상하고 있다.

인터넷을 통한 네트워크 환경은 수많은 자율적인 에이전트들을 포함할 수 있다. 하지만 이러한 에이전트들은 널리 분산되어 있고, 서로 다른 플랫폼을

가지고 있으며, 매우 동적으로 변화한다. 이런 환경에서 작동하는 정보 시스템은 기본적인 몇 가지 문제들을 처리해야만 한다.

우선 인터넷 상에서 널리 쓰여지고 있는 아키텍처인 클라이언트-서버 모델은 너무 제한적이다. 현재 인터넷 정보 서비스들은 사용자가 원하는 대로 새롭고 판단력 있는 요소를 가져다주기 어렵다. 대부분의 에이전트들은 하나의 서버에 의존하기 보다는 서로 상호작용하기를 원한다.

또한 에이전트들의 이질적인 문제를 처리해야만 한다. 이것은 에이전트들이 서로 다른 플랫폼, 서로 다른 데이터 포맷, 서로 다른 정보 서비스의 기능을 가지고 있고 또한 그 서비스에 사용되어지는 서로 다른 표준 시스템을 가지고 있다는 것을 의미한다. 이러한 문제로 한쪽의 정보를 다른 쪽에서 사용할 수 없다면 에이전트 기술은 그 효용 가치가 상당히 떨어진다.

마지막으로 에이전트들은 지능적이어야 하며 인간과 효과적으로 상호작용을 할 수 있어야 한다. 에이전트들은 대화와 타협을 통하여 공동작업을 할 수

*학생회원, 연세대학교 기계공학과 CAD/CAM연구실

**중신회원, 연세대학교 기계공학과

*본 연구는 1996-7년도 교육부 학술연구 조성비(기계공학: ME96-E-19.20)에 의하여 연구되었음.

있어야 하며, 표현적인 통신 언어들을 사용하여 상호 통신할 수 있어야 한다. 또한, 분산되어 있는 지방 자원들을 응용할 수 있고, 인간과의 효과적인 상호작용이 가능해야 한다.

따라서 위와 같은 문제들을 해결하기 위해서는 적절한 에이전트 통신 언어를 선택하고, 에이전트들 간의 새로운 아키텍처를 모델링하여 이를 구현하는 것이 필요하다.

현재 외국의 경우를 살펴볼 때 공학 전반적인 분야에서 인터넷과 에이전트를 이용한 연구들이 활발히 진행되고 있는 실정이나, 국내에서는 아직 연구 및 개발이 미흡한 실정이다.

Finin^[2,3] 등은 에이전트간의 통신 언어와 에이전트를 이용한 프레임워크 개발, 에이전트와 지식 베이스와의 연계, 지능적인 에이전트 개발에 중점을 두고 있으며, Cutkosky^[4,5] 등은 설계에서의 의사 결정 지원, 동시 공학, 협동 설계 등과 같은 엔지니어링 디자인 프로세스에 에이전트 기반 기술을 적용하고 있다. Nishida^[7] 등은 멀티 에이전트 아키텍처를 기반으로 정보 중심의 AI와 지식의 공유/재사용에 관심을 두고 있다.

본 논문에서는 인터넷에서 가장 시각적이고 대중적인 WWW에서도 설계 참여가 가능한 모듈을 개발함으로써 원격지 접속이 자유로운 동시 공학 시스템 개발을 다룬다. 이를 위해서 멀티 에이전트 시스템에서 기존의 에이전트 연결 구조의 문제점과 WWW에서의 새로운 연결 구조를 제시한다. 에이전트 통신 언어로는 KQML(Knowledge Query and Manipulation Language)을 사용하며 몇몇 개발된 에이전트들을 통하여 이들간의 공동 설계 환경 구현을 다룬다.

2. 에이전트

에이전트는 실행하는 동안에 에이전트가 경험하는 환경, 에이전트 자신의 활동, 연결 가능한 다른 에이전트들의 활동들과 이러한 활동들의 결과를 통하여 새로운 지식이 누적될 수 있는 구조로 설계되어야 한다. 이러한 지식들을 바탕으로 에이전트는 자신의 활동 대안을 선정하고, 환경 또는 다른 에이전트들에게 지식을 제공해야 한다. 에이전트는 세분하여 살펴보면 다음과 같은 특징을 가져야 한다.

우선 에이전트는 특수한 영역을 가져야 한다. 에이전트는 일종의 전문가 시스템이라고 할 수 있다. 에이전트는 각각의 특정 영역에서 활동하며, 이 영역과 관련된 지식들을 축적한다. 만약 문제를 수행

하는 동안 자신의 영역과 관련되지 않은 지식이 요구될 때는 그 지식에 전문적인 다른 에이전트들에게 정보를 요청하여 문제를 해결한다.

다음으로 에이전트는 외부 세계와의 인터페이스가 가능해야 한다. 에이전트가 원활히 수행되어지기 위해서는 에이전트가 이용할 수 있는 지식 자원이 충족되어야 한다. 자원이 충족되지 못하면 에이전트의 수행이 중단될 수 있으며 이 때에는 외부 세계와의 인터페이스를 통하여 다른 에이전트들이나 환경, 사용자로부터 필요한 자원의 보충을 받아야 한다. 또한, 에이전트는 소속된 세계를 이해하기 위하여 외부 환경과 정보를 주고 받을 수 있어야 하며, 에이전트 내부의 상황을 반영하기 위하여 내부적으로도 정보의 교환이 원활하게 이루어져야 한다. 더불어 수시로 입력되는 에이전트의 환경 변화에 대한 정보를 최대한 빨리 처리할 수 있도록, 에이전트는 적절한 행동유형을 갖추어 활동을 수행하는데 필요한 정보를 최대한 빨리 발견할 수 있어야 한다.

또한, 에이전트는 기본적으로 목표 달성을 위해 활동을 수행하며, 여러 가지 대안들이 있을 경우 이 중에서 적절한 하나를 선정하여 수행하여야 한다. 에이전트의 각 구성 요소들은 전체적으로 목표 달성에 도움이 되는 활동을 부분적으로 나누어 수행하며 이들을 조합하여 전체적인 목표를 이루어 낸다. 따라서 이러한 조합을 통하여 여러 가지 활동 대안들이 나올 수 있다. 이 경우 에이전트는 자신의 의사 결정 알고리즘이나 외부의 의사결정 지원 시스템의 도움으로 적절한 활동 대안을 선정하고 수행한다. 이와 같은 의사결정 방법은 누적되어 유사한 입력에는 유사한 활동으로 즉시 반응할 수 있는 기능도 갖추어야 한다.

마지막으로 에이전트가 수행한 활동의 결과는 에이전트에 다시 입력되어 변화된 환경을 인식하고 반응하며, 다음 활동에 참고하여야 한다. 또한 에이전트의 활동은 중단없이 계속되어야 한다. 외부세계에 대한 인식, 내부 상황의 고려, 활동대안의 선정, 실행 결과의 인식 등으로 이어지는 에이전트의 활동은 순환 반복되어야 한다.

에이전트의 개념과 특징을 이와 같이 규정할 때에 에이전트 기반 기술은 종래의 전통적인 공정계획 프로그램이나 의사결정 시스템과는 달리 실시간 프로 그래밍에 가까운 개념임을 알 수 있다^[6].

또한 이러한 에이전트 기술은 동시 공학과 밀접한 관계가 있다고 할 수 있다. 동시 공학 환경 하에서 실제적인 또는 가상적인 에이전트들은 제품의 수명기간

에 있어서 제품 개발 시간을 단축하고, 보다 정확하며, 생산 단가를 낮추고 수요자의 만족도를 높여준다.

동시 공학 환경 하의 설계자들이나 엔지니어들은 여러 에이전트들과 더불어 그들의 관점을 설계 초기 단계에서부터 상호 교환함으로써 차후 설계의 수정을 최소화하고 문제 해결의 기반을 미리 마련한다. 또한 네트워크 성능이 발달함에 따라 이러한 공동 작업은 실시간으로 가능하다.

3. 에이전트 통신 언어-KQML

기능적인 에이전트들 사이의 통신을 위해서는 공통적인 구문과 의미, 실제 사용 방법들을 공유해야 한다. 정보와 질의를 표현하는 언어로는 SQL(Structured Query Language), KIF(Knowledge Interchange Format) 등과 같은 것들이 있지만, 어느 하나 보편적으로 받아들여지는 것이 없고 표준화되기에는 이미 각각의 위치에서 확고한 자리 매김을 하고 있어 어려운 상황에 있다.

본 시스템 개발에서는 에이전트 통신 언어로써 KQML을 선택하였다. KQML이란, 다양한 지적 시스템들 사이의 지식과 정보의 교환을 위해 설계된 통신 언어이다^[2,3]. KQML은 대규모의 지식베이스를 공유할 수 있고 재사용이 가능한 구축기술과 방법을 개발시키는 데 목적을 두고 있다. KQML은 일종의 메시지 포맷이지만, 동시에 에이전트간의 실시간 지식 공유를 지원한다. KQML은 지식을 공유하기 위해 지적인 시스템들과 상호작용 하는 응용 프로그램을 위한 언어로써 사용될 수 있다.

KQML은 보통 KIF를 사용하여 각각의 특정 지식에 대해서 서술적으로 기술한다. KQML은 또한 객체 지향의 데이터를 전송하고, 보다 넓은 영역의 정보를 축적하는데 사용되어 진다. 일반적으로 KQML은 질문, 선언, 신뢰, 요구, 획득, 묘사, 제공 등과 같은 정보에 대한 상태를 의사 교환하는데 사용되어 진다. 이러한 에이전트 통신 언어로써 KQML의 유용성은 최근 발표된 몇몇의 논문에서 찾아볼 수 있다^[4].

3.1 KQML의 계층

KQML은 Fig. 1과 같이 세 가지 계층으로 나뉘어진다. 통신 계층에는 송신자, 수신자, 그들의 주소 등 통신과 관련된 여러 가지 요소들을 나열한다. 메시지 계층에는 수행어를 사용하여 메시지의 성질을 정의한다. 내용 계층에는 실제적인 메시지가 들어 있

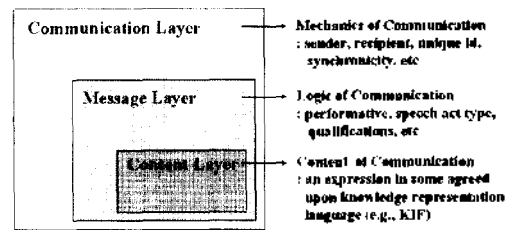


Fig. 1. KQML Layers^[2].

다^[2]. 이 부분에는 KQML, KIF, SQL, ASCII 등 표현 언어라면 어떤 것도 가능하다.

KQML은 메시지 요소들의 리스트를 '(' 로 둘러싸고 있으며, 그 안의 리스트의 첫 번째는 수행어이고, 나머지는 키워드와 그 값의 쌍으로 이루어진다. 대부분 처음 구현되었던 것들은 Common Lisp로 이루어졌기 때문에 현재 구문은 이것과 유사하다.

3.2 수행어 (Performative)

KQML은 확장이 가능한 수행어들에 초점을 두고 있다. 이들을 통하여 에이전트가 다른 에이전트에게 KQML 메시지를 보냈을 때 수신 에이전트가 어떤 동작을 수행하도록 메시지가 의도되었다는 점에서 수행어라고 불린다. 또는 송신 에이전트의 지식을 다른 수신 에이전트들에게 알린다는 점에서 수행어로 정의되기도 한다. 이 수행어는 에이전트들 사이의 상호작용에 있어서 타협과 같은 높은 수준의 통신 모델을 개발하기 위한 기반을 제공한다.

4. 에이전트 통신 구조

4.1 일반적인 에이전트 통신 구조

에이전트들 간의 정보/지식 교환은 메시지 교환을 통하여 이루어 진다. 여러 에이전트들로 이루어진 하나의 시스템을 구축하기 위해 우선 고려하여야 할 사항은 에이전트들 간의 통신 구조를 설정하는 것이다.

여러 개의 에이전트들이 존재할 때, 에이전트들 간의 상호 메시지 전달이라는 측면에서 가장 기본적인 통신 구조는 Fig. 2(a)와 같이 모든 에이전트들 간에 연결을 이루는 것이다. 소수의 에이전트라면 큰 문제는 없으나 에이전트의 수와 사용하는 언어의 수가 증가하면 할수록 연결 선과 번역기는 기하급수적으로 증가하게 된다.

이러한 연결 선의 수를 감소시키기 위해서 사용할 수 있는 통신 구조는 Fig. 2(b)와 같이 중간에 라우터를 두어 메시지 교환을 라우터를 통해서 이루는 것

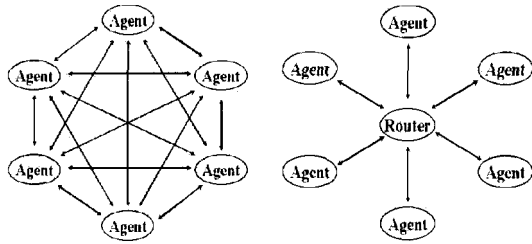


Fig. 2. Basic Connection: (a) Completely connected (b) Star connected.

이다. 이 경우에 연결 선은 감소하게 되지만 메시지 전달의 신뢰성이 조금 떨어지고, 에이전트의 수와 사용하는 언어의 수가 증가하면 할수록 라우터에 많은 부하가 걸리게 된다. 따라서 이러한 문제를 해결할 수 있는 적절한 통신 구조가 필요하다.

4.2 페더레이션(Federation) 구조

이러한 에이전트 통신 구조를 개선하기 위해서 PACT(Palo Alto Collaborative Testbed)에서는 퍼실리테이터(Facilitator)라는 에이전트를 사용하여 Fig. 3과 같은 페더레이션 구조를 제안하였다^[4].

각각의 퍼실리테이터는 자신에 속한 하부 에이전트들 사이의 라우터 역할을 하며, 다른 퍼실리테이터에 속한 에이전트들과의 인터페이스 역할을 담당한다. PACT에서의 퍼실리테이터는 기본적으로 네 가지 목적을 위해서 사용되었다.

- * 신뢰성있는 메시지 전달 계층 제공
- * 적절한 목적지로 메시지 라우팅
- * 자신에 속한 에이전트들에 적당한 언어로 메시지 번역
- * 자신에 속한 에이전트들의 초기화 및 실행 모니터링

퍼실리테이터 간의 메시지 전달은 KQML (Knowledge Query and Manipulation Language)이라는 예

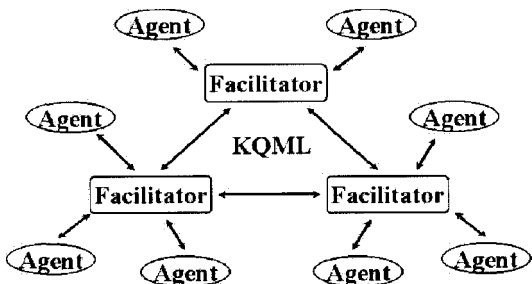


Fig. 3. Federation Architecture.

이전트 통신 언어가 사용되어졌고, 설계 및 프로세스와 관련된 기존의 엔지니어링 도구들로 페더레이션 구조를 구축하고 시나리오를 통하여 이 구조의 효율성을 입증하였다^[4].

4.3 WWW상에서의 에이전트 통신 구조

그러나, PACT에서의 페더레이션 구조에서의 문제점은 기존의 엔지니어링 도구였던 에이전트들이 각각의 시스템 환경에서 그 환경에만 적합하도록 개발되어진 것이다. 즉, 각각의 에이전트들은 다른 환경으로 이식되어질 수 없다는 것이다. 또한, 각각의 에이전트들의 위치가 정적으로 고정되어 있다는 점이다.

에이전트 기술 상의 오랜 문제점이었던 에이전트들의 이질적인 문제는 자바 언어가 개발됨으로 인해 그 실마리를 찾고 있다. 자바 언어로 개발된 응용 프로그램들은 시스템 환경에 무관하게 이식이 가능하다. 또한 자바는 보안 문제로 인해 완벽할 수는 없으나, WWW상에서 프로그램을 실행할 수 있다는 장점이 있다.

현재 국내외에 있는 여러 연구 기관에서 자바를 이용하여 에이전트들을 개발하고 있다. WWW과 상관없는 응용 프로그램의 경우에는 자체적으로 서버 소켓을 생성할 수 있으므로 큰 문제점은 없다. 하지만 넷스케이프나 마이크로 소프트의 Internet Explorer와 같은 웹 브라우저를 통하여 자바 프로그램인 애플릿을 실행하였을 때, 애플릿은 서버 소켓을 생성할 수 없고, 단지 이것을 생성한 HTTP 서버만 통신할 수 있다는 보안 제약이 있다. 따라서 현재 구현되고 있는 WWW을 이용한 에이전트 통신 구조는 Star 구조의 형태를 띠고 있다.

본 연구 개발에서는 스탠포드 대학의 CDR(Center for Design Research)에서 개발한 JATLite 자바 API를 이용하고 있다^[1]. JATLite는 에이전트 개발을 위한 기본적인 자바 클래스 라이브러리를 제공한다.

현재 JATLite는 TCP/IP와 KQML을 이용하여 구현을 하고 있지만, 또 다른 에이전트들의 통신을 위해서 통신 언어와 관련된 추상 계층과 기본 통신 계층을 제공한다. 또한, 이 두 계층을 바탕으로 하여 실제적으로 메시지를 표현하고 해석하고 처리하는 KQML 계층과 라우터 계층, 프로토콜 계층을 제공한다.

4.4 DCM(Dynamic Connection Manager)을 통한 페더레이션 구조

본 연구 개발에서는 자바 언어를 이용하여 에이전

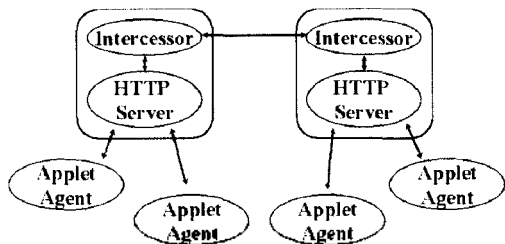


Fig. 4. Intercessor Connection in the Web.

트들을 개발하였으며, WWW상에서의 Star 구조를 개선하고 에이전트 연결 구조를 확장하기 위하여 퍼실리테이터와 유사한 개념의 인터세서(Intercessor)란 에이전트를 제시하고, 이것을 이용하여 WWW상에서의 페더레이션 구조를 이룩하였다. Fig. 4와 같이 각각의 HTTP 서버에 인터세서를 두어 HTTP 서버로부터 다운로드 된 애플릿 에이전트들은 그 서버에 있는 인터세서와 연결된다. 또한 각각의 인터세서들은 실행될 때 다른 인터세서들과 연결된다. 이러한 구조는 어떤 HTTP 서버에서 다운로드 된 애플릿 에이전트와 다른 HTTP 서버에서 다운로드 된 애플릿 에이전트 사이의 메시지 교환을 가능하게 한다.

인터세서의 주 기능을 요약하면 다음과 같다.

- * 메시지 중개
- * 메시지 임시 저장소
- * 부속 에이전트들의 실행 모니터링
- * 공동 설계 데이터/모델 저장소
- * ANS (에이전트 네임 서비스)

하지만, 이러한 구조에서의 문제점은 WWW상에서의 에이전트들이 매우 동적으로 변화한다는 것이다. 각각의 에이전트들은 네트워크 연결을 통하여 각각의 WWW 브라우저로 다운로드 되기 때문에 네트워크 상에서 어떤 한 위치에 고정되어 있는 것이 아니라 수시로 그 위치가 바뀌게 된다. 각각의 인터세서에 연결 관리 기능을 두어 자신에 속한 에이전트들은 관리가 가능하지만, 다른 인터세서에 속한 에이전트들에 대해서 관리가 불가능하다.

예를 들어, "A"라는 에이전트가 "B"라는 에이전트에게로 메시지를 보낼 때, 일단 이 메시지는 "A"를 관리하고 있는 "가"라는 인터세서에게로 보내진다. "가"는 자신이 관리하고 있는 에이전트들 중에서 "B"를 찾아보고 없으면, "B"를 관리하고 있는 인터세서에게 메시지를 보내어 이 인터세서가 "B"에게 메시지를 전달해야 한다. 문제는 어느 인터세서가 "B"를 관리하고 있는가 라는 점이다.

이 문제를 해결하는 방법은 Fig. 5와 같이 일단 임

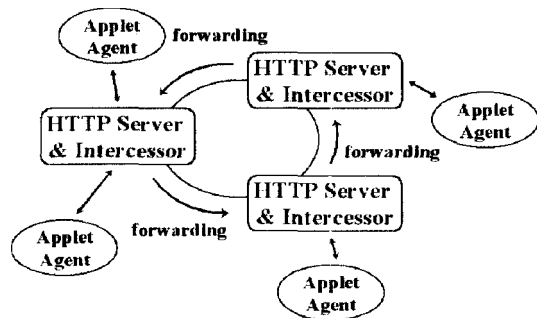


Fig. 5. Federation with Ring Connection.

의 다른 인터세서에게 메시지를 전달하는 것이다. 메시지를 받은 인터세서는 자신이 관리하고 있는 에이전트들 중에 "B"가 있으면 "B"에게 메시지를 전달하고 없으면 또 다른 인터세서에게 메시지를 전달하는 것이다.

하지만, 이 경우의 문제점은 "B"를 관리하는 인터세서가 없거나, 인터세서 간의 전달 구조에 이상이 생길 경우 메시지가 무한히 순환하게 되는 손실을 야기시킨다.

또 다른 문제 해결 방법은 Fig. 6과 같이 일단 모든 인터세서에게 "B"를 관리하고 있는 가를 알아보는 것이다. 자신이 관리하고 있다는 메시지를 보낸 인터세서가 있다면 그 인터세서에게 메시지를 전달하고, 연결된 인터세서들이 모두 관리하고 있지 않다는 메시지를 보내면 처음 메시지를 보낸 "A"에게 "B"는 등록된 에이전트가 아니라라는 예러 메시지를 보낸다.

하지만, 이 방법은 에이전트 구조가 확장되어 인터세서의 수가 많아지게 되면 각각의 인터세서에게 메시지를 보내고 이에 대한 응답을 받아야 하는 수가 많아지게 되고, 마지막 인터세서에게서 응답을 받을

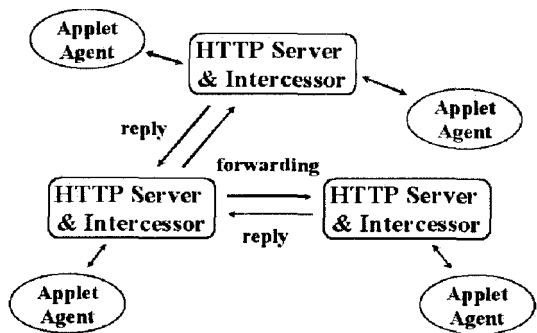


Fig. 6. Federation with Complete Connection.

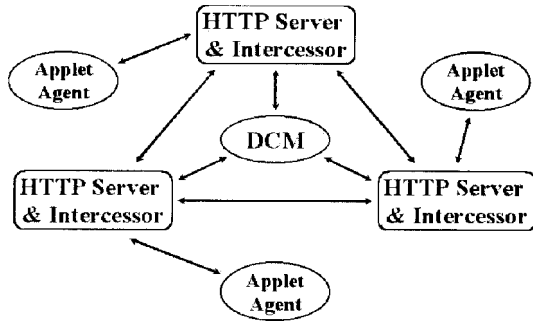


Fig. 7. Federation with DCM.

때까지 메시지 전달을 기다려야 한다는 단점이 있다.

본 연구에서는 위와 같은 문제를 해결하기 위해서 Fig. 7과 같이 DCM이라는 에이전트를 두었다. 인터세서는 실행될 때 DCM과 연결하여 자신이 관리하는 에이전트들에 대한 연결 정보를 DCM에게 알려주며, 새로운 에이전트가 등록하게 되면 이 사실을 DCM에게 전달한다. 따라서 DCM은 자신과 연결된 각각의 인터세서들에 대하여 그들이 관리하고 있는 에이전트들에 대한 연결 정보를 유지 관리하며, 인터세서들이 다른 인터세서들의 에이전트 연결 정보를 요청할 때 이를 알려준다.

다른 인터세서에게 속해있는 에이전트에게 보내야 할 메시지를 받은 인터세서는 우선 DCM에게서 그 에이전트를 관리하는 인터세서를 알아내어 그 인터세서에게 메시지를 전달하는 것이다. 즉, DCM은 에이전트 연결 구조를 확장하기 위해서 여러 개의 인터세서들을 들 때 일어날 수 있는 메시지 전달의 불확실성을 제거하기 위한 특수 목적의 에이전트이다.

5. 시스템 구현

우리는 위에서 설명한 아키텍처를 구현하고 이를 검증하기 위해 간단한 예제 시나리오를 구성하여 Fig. 8과 같이 적용하였다. 이 연결 구조는 동적으로 바뀔 수 있다. 에이전트들은 각각 그들이 속한 인터세서들을 중심으로 메시지 교환을 통하여 그들의 작업을 수행한다. 인터세서들은 DCM과 연결하여 다른 인터세서에 속한 에이전트들에 대한 정보를 수시로 교환한다.

적용된 시나리오는 특정 제품에 사용되는 모터를 결정하기 위한 과정이다. 에이전트들은 각각 서로 다른 역할을 맡고 있으며, 인간과 함께 작업을 수행

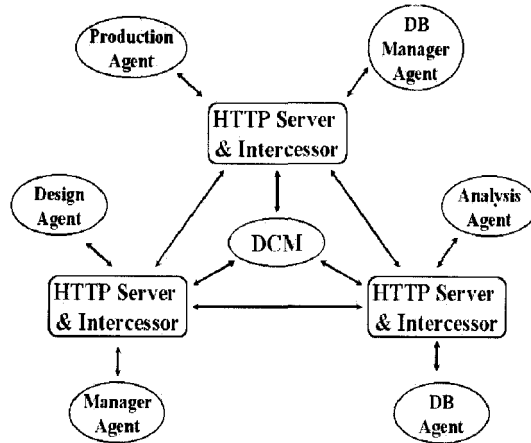


Fig. 8. A Whole System Connection Structure.

하는 에이전트들과 컴퓨터 단독으로 모든 일을 처리하는 서비스 에이전트들로 구성된다. 에이전트들의 이질적인 문제 해결을 위해서 시나리오의 실행은 각각 Windows 95/NT, UNIX에서 자바 애플리케이션들과 애플릿들로 구현되었다.

인터세서, DCM 및 서비스 에이전트들은 시스템 내부에서 각각 애플리케이션으로서 실행되어 진다. Fig. 9는 인터세서와 DCM의 실행과정을 문자로 모니터링한 것을 보여준다. 서비스 에이전트들도 시스템 내부에서 각각 애플리케이션으로 실행된다. 이들은 통신 모듈에서 인터세서에게 등록된 에이전트들의 목록을 얻음으로써 이들의 실행 여부를 알 수 있다.

에이전트들 중에는 서비스 에이전트와 달리 사람의 능력을 필요로 하는 에이전트들도 있다. 이러한 에이전트들은 메시지의 종류에 따라 에이전트 스스로 해결하기도 하고, 결정적인 중요한 선택은 사람의 능력에게 맡기게 된다. 또한 사람으로 하여금 자신의 지식이나 경험을 다른 에이전트들에게 전할 수도 있다. 따라서 사용자와의 인터페이스 역할을 할 수 있는 에이전트가 필요하며, WWW상에서 이러한 역할을 담당하는 것이 애플릿 에이전트이다.

애플릿 에이전트는 통신 모듈과 CAD 모듈로 구성되며, 통신 모듈 한 가지로도 에이전트 설정이 가능하다. Fig. 10은 CAD 모듈을 보여주고 있다. 기본적인 객체로서 Box, Cylinder, Cone, Sphere를 생성할 수 있으며, 2차원 형상의 Extrusion 및 회전체 생성이 가능하다. Zoom-in, Zoom-out, Rotation을 통하여 모델을 살펴볼 수 있으며, 삭제와 수정이 가능하며, 애플리케이션이나 애플릿 모두 자신의 파일 시스템에 저장하거나 자신의 시스템에서 불러오는 것

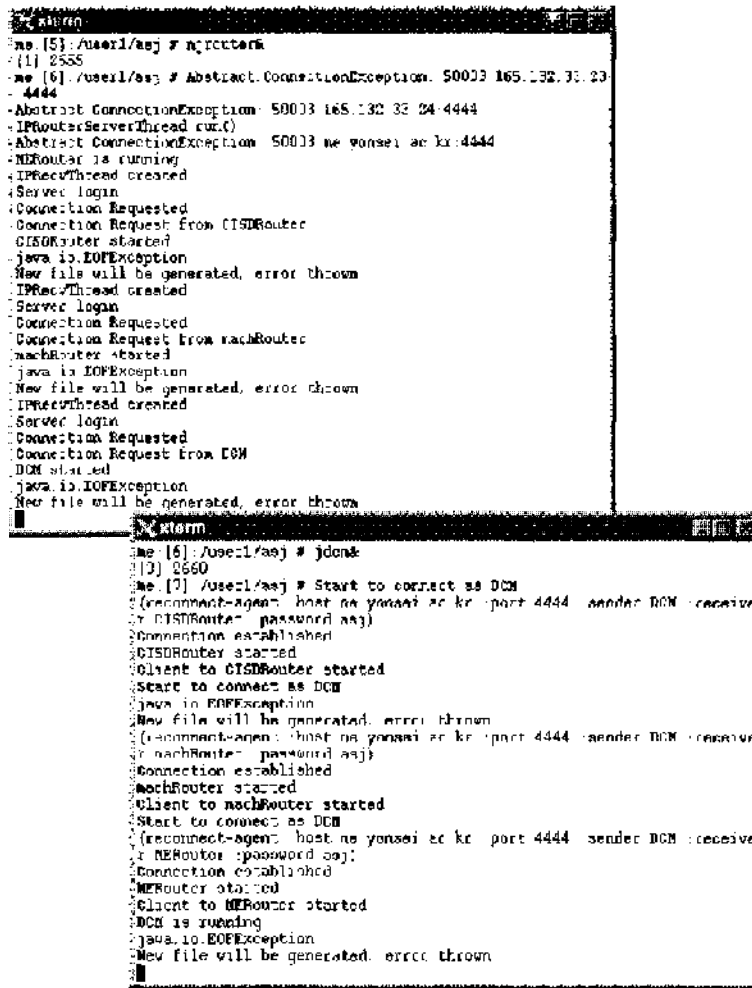


Fig. 9. Internal Execution Screen of Intercessor and DCM.

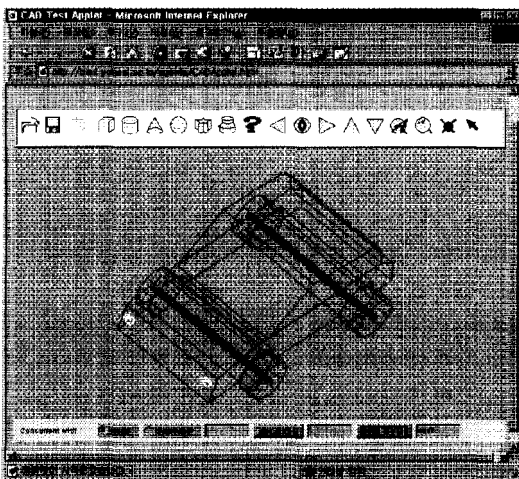


Fig. 10. CAD Module.

이 가능하다.

또한 설계한 모델을 인터세서의 공동 설계 데이터/모델 저장소에 모델을 임시 저장함으로써 다른 에이전트가 그 모델을 불러오는 것이 가능하다.

연결 아이콘을 선택하면 Fig. 11과 같은 통신 모듈이 생성된다. 통신 모듈은 인터세서와의 연결을 위한 모듈이다. 등록과 연결에 해당하는 KQML 메시지를 버튼을 이용하여 자동적으로 생성하고 인터세서에 전달한다. 그 밖의 몇 가지 메시지에 대해서도 적절한 KQML 메시지를 생성하고 전달한다.

기본적으로 에이전트를 등록시킬 수 있는 창이 제공되며, 다른 에이전트로부터의 메시지를 볼 수 있는 창과 다른 에이전트에게로 메시지를 보낼 수 있는 창이 있다. 애플릿 에이전트의 경우에는 KQML 구문에 맞는 모든 메시지를 메시지 창을 이용하여

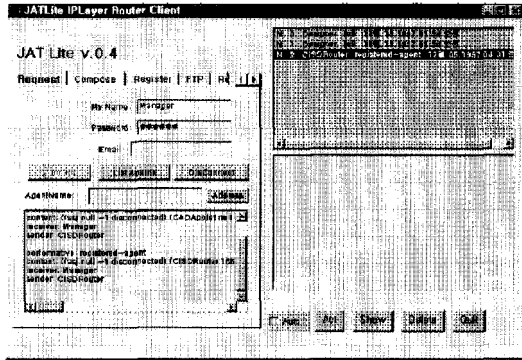


Fig. 11. Communication Module.

보내거나 볼 수 있다. 자주 사용되는 메시지 전달은 버튼을 사용하여 처리한다.

받은 메시지들의 리스트가 나열되는 창이 있으며, 창안에 나열된 메시지를 선택하면 메시지 상자를 통하여 그 내용을 볼 수 있다. 메시지 전달에 문제가 발생시에는 에러 메시지 상자를 통하여 에러의 원인을 알 수 있다.

이 CAD 모듈의 가장 큰 특징은 공동 설계가 가능하다는 것이다. 떨어져 있는 두 에이전트가 WWW 브라우저를 통하여 함께 모델을 설계할 수 있도록 하였다. 한 에이전트에서 하나의 객체를 추가, 변경, 삭제하면 이 정보가 다른 에이전트에게 전달되어 다른 에이전트의 CAD 모듈에서도 같은 상황이 전개

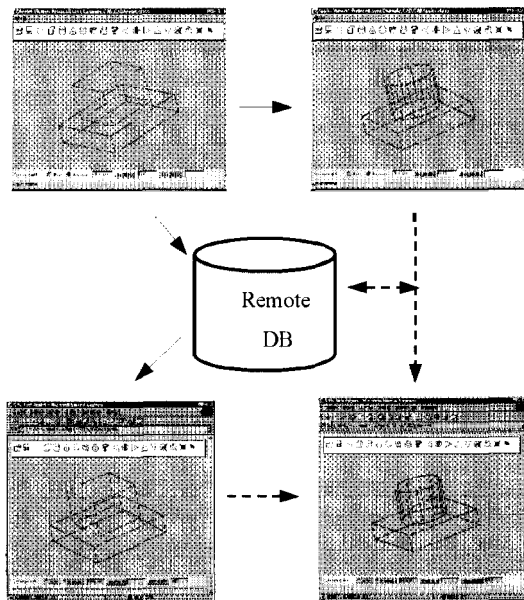


Fig. 12. Collaborative Design of CAD Modules.

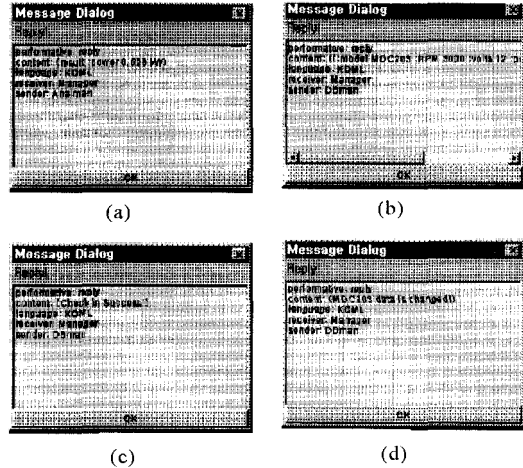


Fig. 13. Reply Messages from Service Agents.

된다. 따라서 네트워크의 전송 속도에 좌우되기는 하지만 거의 실시간으로 공동 설계가 가능하다.

Fig. 12는 CAD 모듈과 인터페이스의 모델 저장소를 이용하여 두 사용자가 공동으로 설계하는 것을 나타낸다. 시나리오 구현에서 설계 에이전트는 CAD 모듈을 이용하여 설계를 하고 매니저 에이전트와 Fig. 12와 같은 방법으로 함께 수정 작업을 하게 된다. 매니저 에이전트는 서비스 에이전트들에게 요청하여 Fig. 13(a), (b), (c)와 같은 결과를 얻어낸다. 매니저 에이전트가 부품 정보 서비스 에이전트에게 어떤 부품에 대하여 체크인을 하면 그 부품에 대한 정보가 바뀌었을 때, 매니저 에이전트에게 이 사실을 Fig. 13(d)와 같이 자동적으로 알려준다.

6. 결 론

본 연구에서는 기존의 에이전트 연결 구조의 문제점을 지적하였으며, 점차 그 중요성과 사용도가 증가하는 WWW상에서의 새로운 에이전트 연결 구조를 제시하고, 자바 언어를 이용하여 이것을 구현하였다.

네트워크 기술이 발달하고 전송속도가 빨라짐에 따라 기존의 텍스트 기반의 인터넷 사용에서 점차 WWW 브라우저 등을 통한 시각적이고 실용적인 인터넷 사용이 요구되어지고 있다. WWW 브라우저에서 실행되는 애플릿은 이러한 요구 사항들을 만족시켜줄 뿐만 아니라 하나의 에이전트로서 동시 공학 시스템을 지원하는 모듈이 될 수 있다.

본 연구에서는 자바 언어를 사용하여 기존의 에이

전트 기술 상의 문제점이었던 시스템 개발의 이질적인 문제를 해결하고, WWW상에서의 에이전트 연결 구조를 위해서 인터세서와 DCM이라는 에이전트를 제시하고 구현하였으며, 이것들을 이용하여 새로운 연결 구조를 형성하고 시나리오를 통하여 에이전트들의 공동 설계의 가능성을 검증하였다. 여기에서 DCM은 단순한 클라이언트 에이전트들의 연결 관리가 아니라 일종의 서버 역할을 할 수 있는 인터세서들 간의 연결을 관리한다. 즉, 분산 환경 간의 연결을 시도함으로써 보다 많은 정보와 서비스를 공유하도록 한다.

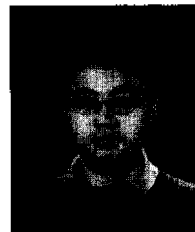
본 연구에서 얻어진 결론을 종합하면 WWW상에서 실행 가능한 에이전트들로 시스템을 구축하기 위해서는 기존의 연결 구조를 개선해야 하고, 본 연구에서 제시된 인터세서와 DCM은 WWW상에서의 에이전트 연결 구조를 가능하게 하며, 또한 연결 구조의 확장을 가능하게 해준다. 또한 본 연구를 통하여 볼 때, WWW을 이용한 동시공학 시스템 구축이 가능하다.

향후 연구 과제로는 기존의 엔지니어링 도구들과의 인터페이스를 구축하여 자바 에이전트들과의 연계를 형성하고, 에이전트가 좀 더 자율적일 수 있도록 배이스를 확장하는 것이 요구된다. 또한 각각의 전문 영역에 적합한 다양한 애플릿 에이전트의 개발과 실용적인 측면에서 보다 상세하고 전문적인 시나리오의 적용이 요구된다.

참고문헌

1. "JATLite", <http://java.stanford.edu/>.
2. Finin, T., McKay, D. and Fritzson, R., *An Overview of KQML: A Knowledge Query and Manipulation Language*, Technical Report, Computer Science Department, University of Maryland, Mar. 1992.
3. Mayfield, J., Labrou, Y. and Finin, T., "Evaluation of KQML as an Agent Communication Language", *Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages*, Vol. 2, pp. 347-360, 1995.

4. Cutkosky, M., Genesereth, M., et al., "PACT: An Experiment in Integrating Concurrent Engineering Systems", *IEEE Computer Special Issue on Computer Supported Concurrent Engineering*, Vol. 26, pp. 28-37, Jan. 1993.
5. Petrie, C., Jeon, H. and Cukosky, M., "Combining Constraint Propagation and Backtracking for Distributed Engineering", *Working Notes of the ECAI-96 workshop on Non-Standard Constraint Processing*, Aug. 1996.
6. Frost, H. and Cutkosky, M., "Design for Manufacturability via Agent Interaction", *Proceedings of the 1996 ASME Computers in Engineering Conference*, pp. 1-8, Aug. 1996.
7. Nishida, T. and Takeda, H., "Towards the Knowledgeable Community", *Proceedings International Conference on Building and Sharing of Very-Large Scale Knowledge Bases '93*, pp. 157-166, Dec. 1993



안 상 준

1996년 연세대학교 기계공학과 졸업
 1998년 연세대학교 기계공학과 대학원 졸업
 1998년 ~ 현재 삼성자동차 중앙연구소 주임 연구원
 관심분야: 지능형 CAD, 네트워크 기반 동시 설계시스템, CAD/CAM



이 수 홍

1981년 서울대학교 기계공학과 학사
 1983년 서울대학교 기계설계학과 석사
 1991년 Stanford대학 Design Division Concurrent Engineering 전공, 박사
 1991년 ~ 1992년 Lockheed Missile and Space Co. Cable Harness Design System개발 Post-Doc.
 1983년 ~ 1994년 KIMM CAD/CAM실, 선임 연구원
 1994년 ~ 현재 연세대학교 기계공학과, 부교수
 관심분야: 동시공학설계, 지식기반시스템 설계, DFM