에이전트 시스템 개발도구에 관한 연구

이 광 로[†] · 박 상 규[†] · 장 명 욱[†] · 민 병 의[†] · 황 승 구[††]

요 약

에이전트 시스템은 자발성, 자율성, 사회성, 반응성을 갖는 독립된 프로그램인 에이전트를 조합하여 구성되는 시스템으로, 일반 사용자에게 편리하고 자연스러운 메타포를 제공한다. 그러나 개발자 측면에서는 에이전트 시스템에서 요구하는 각종 기능 및 제약규칙을 따라야 하기 때문에 여러 가지 문제점과 어려움이 따른다. 특히 에이전트 시스템이 표준화되어 있지않은 현상태에서 상호 교류 없이 만들어진 에이전트 시스템은 상호 활용이나 에이전트 간의 정보교환이 거의 불가능하다. 본 논문에서는 에이전트 시스템을 개발할 때 발생하는 문제점들을 해결하기 위해 에이전트 개발도구가 갖추어야 할 요구사항을 정리하고, 이러한 요구사항을 바탕으로 개발한 프로토타입 시스템인 에이전트 개발도구(ADT:Agent Development Tools)를 소개한다. 에이전트 개발도구의 개발 목적은 에이전트 시스템에 대하여 전문가이든 비전문가이든 손쉽게 에이전트화 할 수 있고, 에이전트 시스템 관련 자원을 효율적으로 관리할 수 있게 하는 것이다. 따라서 ADT를 사용하면 누구나 쉽게 에이전트 시스템을 확장할 수 있고, 또한 높은 질의 서비스를 받을 수 있다.

A Study on Tools for Agent System Development

Gowang-Lo Lee[†] · Sang-Kyu Park[†] · Myeong-Wuk Jang[†] · Byung-Eui Min[†] · Seung-Ku Wang[††]

ABSTRACT

An agent system is consisted of agents that have the following properties: autonomy, social ability, reactivity, pro-activeness. It provides a good environment to an end-user. Therefore, an end-user can feel more natural and comfortable in using a computer system. However, a system developer will have difficulty because he(or she) has to follow the functions and the specification which are generally required for an agent system. Moreover, it is hard to use resources mutually and share information between agent systems that are solely developed without satisfying the standardization. In this paper, we describe the requirements for and agent development tools (ADT). We also introduce the ADT that is developed to efficiently resolve the problems occurred in developing an agent system. The main properties of the ADT are: either expert or non-expert can easily interface an application to existing agent systems and effectively manage the resources that are related to the agent systems. Therefore, one can easily expand an agent system and achieve high quality of services with our ADT.

1. 서 론

최근 에이전트 개념을 도입한 연구가 각 분야에서 활발히 연구되고 있으며, 에이전트 시스템 분야에서는 구조[1, 2, 3, 4, 5, 6], 언어[7, 8], 알고리즘[9, 10], 응용영역[9, 11, 12, 13], 개발도구[14, 15, 16, 17] 등에서 진행중이다. 에이전트 시스템은 자발성, 자율성, 사회성, 반응성을 갖는 독립된 프로그램인 에이전트를 조

† 정 회 원:한국전자통신연구원 인공지능연구실
†† 정 회 원:한국전자통신연구원 멀티미디어연구부 부장
논문접수:1996년 8월 29일, 심사완료:1996년 11월 13일

합하여 구성되는 시스템[12]으로, 일반 사용자에게 편리하고 자연스러운 메타포를 제공한다. 이것은 미래 사회가 인터넷을 기반으로 전자상거래, 원거리 교육, 원거리 영상통신 등 많은 서비스들을 제공할 때 컴퓨터 사용에 익숙하지않은 많은 사용자의 어려움을 극복해 줄 수 있다. 또한 사용자가 반드시 처리해야할 일 중 데이터의 다운로드와 같이 시간이 많이 걸리는 일, 웹사이트와 같이 정보가 많이 흩어져있는 곳에서 필요한 정보를 수집하는 일, 호텔 예약이나 교통편 예약과 같이 한정된 정보를 이용하여 정해진 작업을 수행하는 일 등과 같은 부가가치가 적으면서 많은 노력(cost)을 요하는 작업을 대행해 줄 수 있다. 따라서 사용자는 부가가치가 높은 일에 좀더 많은 시간을 보낼 수 있다. 이와 같이 에이전트 시스템의 도입은 사회적/경제적 측면으로 많은 혜택을 가져다 줄 것이고, 에이전트 시스템의 확산도 매우 급속도로 진행되리라 예측된다. 특히, 에이전트 시스템을 개발할 때 상호 고려 없이 독립적으로 만들어져 사용되고 있는 많은 응용 소프트웨어를 그대로 사용할 수 있기 때문에 소프트웨어의 재사용성에서 매우 유용하다.

그러나 일반 개발자가 기존의 응용 프로그램이나 새로 개발하려는 응용 프로그램의 기능향상을 위해 에이전트 프로그램을 작성하는 것은 많은 관련 지식과 시간이 요구된다. 현재, 새로운 에이전트를 생성하기 위해서는 개발하고자 하는 응용영역에 대한 지식뿐만 아니라 에이전트의 개념, 에이전트 시스템 구조, 멀티모달 처리기술(자연어처리, 음성인식, 펜인식) 등과 같은 에이전트 시스템에 대한 지식과 에이전트 시스템과 응용 프로그램 간의 인터페이스를 상세히 알고 있어야 한다. 그러나 이러한 조건을 만족하는 개발자는 그다지 많지않으며 이러한 조건을 갖추기 위해서는 많은 노력과 시간이 요구된다.

본 논문에서는 에이전트 시스템을 개발할 때 발생하는 문제점들을 해결하기 위해 에이전트 개발도구가 갖추어야 할 요구사항을 정리하고, 이러한 요구사항을 바탕으로 OAA(Open Agent Architecture) 시스템[1, 3, 5]을 목표 시스템(target system)으로 한 에이전트 개발도구의 프로토타입인 ADT(Agent Development Tools)를 소개한다. OAA 시스템은 기존의 그래픽 사용자 인터페이스의 차원을 넘어서 사용자의 작업을 대행해주는 지적인 사용자 인터페이스로서 블랙보드의 개념을 바탕으로 한 멀티에이전트 구조를 갖는다. 특히, 개발자에게 기존의 응용 프로그램을 이용하여 새로운 서비스를 쉽게 제공할 수 있도록 플러그 & 플레이(plug & play)를 지원하는 개방형 시스템이다. OAA 시스템은 현재 개발된 에이전트 시스템 중 에이전트가 가져야 할 특성인 자발성, 자율성, 사회성, 반응성 등을 모두 가진 시스템으로 확장성이 뛰어나고, 시스템간의 이형성을 지원하는 에이전트 통신언어를 갖춘 우수한 시스템이다[1, 5].

본 논문의 구성은 1장에서 서론을 기술하고, 2장에서는 에이전트 개발도구의 요구사항을 기술하고, 3장에서는 ADT의 목표 시스템인 OAA 시스템에 대하여 기술하고, 4장에서 에이전트 개발도구인 ADT에 대하여 기술하고, 5장에서 결론을 기술한다.

2. 에이전트 개발도구의 요구사항

일반 사용자에게 보다 편리한 사용자 인터페이스 환경을 제공하기 위해서는 현재의 윈도우즈 기반 사용자 인터페이스의 차원을 넘어서 사용자의 작업을 대행해 줄 수 있는 에이전트 시스템이 제공되어야 한다. 또한, 에이전트 시스템의 서비스 확장과 사용보급을 위하여 응용 소프트웨어를 에이전트화 할 때 특정 개발자에게 한정된 것이 아니라 일반 응용 소프트웨어 개발자도 쉽게 응용 소프트웨어를 에이전트화 할 수 있는 에이전트 개발도구가 요구된다. 이러한 기능을 효율적으로 지원하기 위해 에이전트 개발도구가 갖추어야 될 요구사항을 다음과 같이 다섯 가지 측면에서 제안한다.

(1) 확장성

확장성의 문제는 크게 에이전트 시스템 기반구조의 확장과 에이전트 개발도구의 확장으로 볼 수 있다. 에이전트 시스템 기반구조의 확장을 보면, 에이전트 시스템의 기능이나 서비스를 확장하기 위해서는 새로운 에이전트를 개발하거나 기존의 에이전트를 수정해야만 한다. 이러한 작업을 수행하기 위해서는 에이전트 시스템 기반구조에서 요구하는 각종 기능 및 제약규칙에 따라 에이전트를 생성하거나 수정하여, 이것을 기반구조에 통합시켜야 한다. 따라서 에이전트 개발도구는 에이전트 시스템의 기능 확장을 위

해 필요한 개별 에이전트 생성기능, 멀티모달 추가 기능, 에이전트 시스템 구축 기능 등을 쉽게 확장할 수 있는 환경을 지원해야 한다.

에이전트 개발도구의 확장을 보면, 에이전트 시스템은 사용자의 편의와 시스템의 효율을 위하여 시스템의 구조와 패러다임이 지속적으로 변할 수 있다. 예를 들면, 에이전트 구조(agent framework) 설계시 고려하지않은 특정 서비스를 지원하기 위한 새로운 지식처리모듈 추가, 에이전트 시스템 구조에 에이전트의 이동성을 추가하기 위한 모듈 추가, 가상현실세계 내에 에이전트 작업환경을 제공하기 위한 모듈 추가 등이다. 따라서 에이전트 시스템의 구조확장이나 새로운 파라다임을 확장하려 할 때 이것을 고려하지 않고 설계 구현된 에이전트 개발도구는 다시 만들거나 기존의 프로그램을 저수준에서 수정되어야 한다. 따라서 이러한 문제를 해결하기 위해서는 에이전트 개발도구를 개발할 때 각 기능을 지원하는 모듈이 독립적으로 작성되어야 하며, 에이전트 개발도구의 구조 또한 동적이고 개방적이어서 추가하는 새로운 모듈을 쉽게 통합할 수 있어야 한다.

(2) 이식성

에이전트 시스템 내의 각 에이전트는 시스템 간의 이형성을 지원하기 위하여 에이전트 통신언어를 이용하여 서로 다른 환경에 있는 에이전트와 상호 협력하면서 필요한 작업을 수행한다. 그러나 사용환경(H/W 플랫폼이나 OS)이 바뀌면 기존에 사용하던 환경과 다르기 때문에 이전에 사용하던 에이전트 시스템이 정상적으로 동작하지 않아 새로운 환경에 적절하게 에이전트 시스템을 재컴파일 하거나 재작성 해야하는 문제가 발생한다. 또한, 대부분의 개발자는 개인적인 작업의 효율을 위해 자기만의 고유한 언어와 개발환경을 구축하여 사용한다. 그러므로 에이전트 시스템을 개발하기 위해 특정 환경과 개발언어를 제한할 경우 개발이 비효율적인 뿐만 아니라 많은 문제를 발생시킬 수 있다. 따라서 에이전트 개발도구는 에이전트 시스템의 사용환경이나 개발언어에 종속되지않는 다양한 개발환경을 수용할 수 있어야 한다.

(3) 공유성

분산환경에서 에이전트 시스템으로부터 다양한 서비스를 지원 받기 위해서는 많은 응용 에이전트를 개발해야 한다. 이때 필요한 에이전트를 개별적으로 개발해서 관리한다면 같은 서비스를 지원하는 에이전트를 중복해서 개발하는 경우가 많이 생길 것이다. 이러한 문제를 해결하기 위한 방법으로 에이전트 개발자들이 글로벌 서버를 이용하여 에이전트 관련 정보를 공유하는 것이다. 즉, 에이전트 개발자들이 서로 정보를 공유할 수 있는 글로벌 서버에 특정영역을 정하고, 새로운 에이전트를 만들면 이곳에 옮겨서 정보를 공유한다. 그러면 에이전트를 새로 개발하려고 할 경우, 먼저 특정영역에 저장된 에이전트 관련 정보를 보고 원하는 에이전트가 이미 있으면, 그것을 다시 만들 필요 없이 해당 에이전트 개발자의 동의를 얻어 가져다가 사용할 수 있어 마치 자신이 만든 것처럼 사용한다. 만일 원하는 에이전트가 없다면, 이와 유사한 에이전트 관련 정보를 참조하여 만든다면 처음부터 설계하여 만드는 것보다 훨씬 개발이 쉽고 효율적일 것이다. 따라서 에이전트 개발 도구는 개발환경에 종속되지않고 언제 어디서든지 에이전트 개발 관련 정보를 열람하고 재사용할 수 있는 환경을 제공해야 한다.

(4) 관리성

에이전트 수가 많아 짐에 따라 에이전트 관련 자원을 효율적으로 관리해주는 기능이 요구된다. 특히, 다른 개발자와의 에이전트 관련 정보의 공유에 따라 에이전트 관련 자원을 효율적으로 관리해주는 기능이 요구된다. 이러한 에이전트 자원은 에이전트 등록에 필요한 정보, 다른 개발자와 에이전트 공유를 위한 정보, 에이전트 자연어 인터페이스 관련 정보, 에이전트 시스템 구축에 관련된 정보 등 다양한 데이터로 이루어진다. 이러한 데이터는 에이전트 내의 서로 다른 모듈에서 독립적으로 사용된다. 그러나 각 데이터 간에는 밀접한 연관관계를 가지고 있다. 즉 이러한 에이전트 관련 데이터는 자신만이 열람하면서 에이전트 시스템에 사용되는 것과 다른 개발자와 공유하면서 사용되는 것으로 나누어진다. 에이전트 시스템이 복잡해지고 다양한 에이전트가 개발됨에 따라 이러한 다양한 데이터를 일관된 인터페이스를 통하여 통합관리할 수 있어야 한다. 또한 에이전트의 공유성 지원을 위해 에이전트 관련 정보가 로컬영역 뿐만 아

니라 원거리의 특정 영역에도 존재하기 때문에 이러한 정보도 통합관리할 수 있어야 한다.

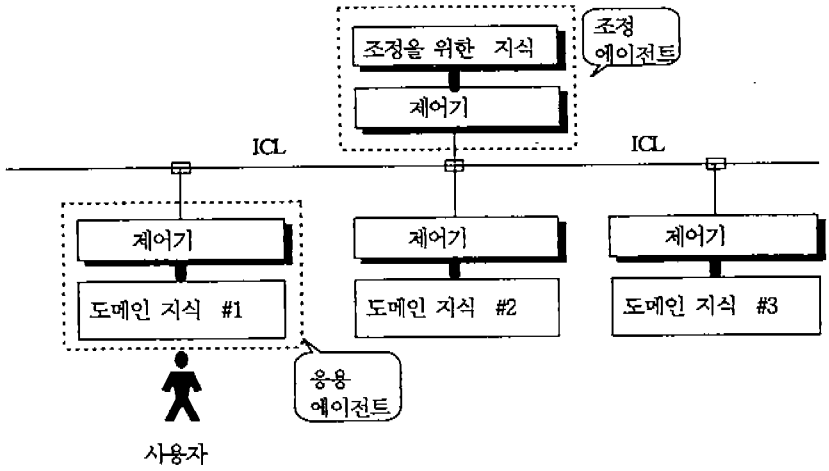
(5) 조작성

새로운 에이전트를 생성하기 위해서는 개발하고자 하는 응용영역에 대한 지식 뿐만 아니라 에이전트의 개념, 에이전트 시스템 구조, 멀티모달 처리기술 등과 같은 에이전트 시스템에 대한 지식과 에이전트 시스템과 응용 프로그램간의 인터페이스를 상세히 알고 있어야 한다. 그러나 이러한 조건을 만족하는 개발자는 그다지 많지않으며 이러한 조건을 갖추기 위해서는 많은 노력과 시간이 요구된다. 따라서 에이전트 개발도구는 에이전트 기반구조에서 요구하는 규정에 따라 많은 응용 소프트웨어를 에이전트화 할 때 응용영역에 대한 지식 외에 특별한 추가 지식을 요구하거나 조작이 어려워 작업의 효율을 저하시켜서는 안된다. 즉 에이전트 전문 개발자와 비전문 개발자 모두에게 조작이 편리하고, 생산성을 향상시킬 수 있어야 한다. 예를 들면, 기반구조에서 요구하는 정보와 지식의 습득을 개발자에게 간단한 질문을 통해 얻거나 필요한 지식을 예제를 제시한 후 개발자에게 선택하게 하여 얻는다. 기반구조에서 공통적으로 필요로 하는 기능 및 데이터를 에이전트 코드형판에 삽입하여 제공함으로써 사용자는 필요한 기능을 코드형판에서 제공하는 코드에 삽입만 하면 에이전트 시스템에서 사용할 수 있는 에이전트를 만들 수 있게 한다.

3. OAA(Open Agent Architecture) 시스템

OAA 시스템은 Schwartz가 제안한 FLiPSiDE 시스템[2]으로부터 간접적인 영향을 받아 블랙보드 구조를 기본으로 하고있다. OAA는 하나의 조정 에이전트와 둘 이상의 응용 에이전트로 이루어진 분산환경과 플러그 & 플레이를 바탕으로 한 동적인 구조를 갖는다((그림 1) 참조). 에이전트는 그 역할에 따라 분류가 되는데 조정 에이전트는 응용 에이전트들 간의 교류를 위한 장소 및 조정기능을 제공한다. 이와 대조적으로 응용 에이전트는 특정한 도메인의 작업을 전문적으로 처리하는 역할을 맡는다. OAA 시스템에서는 에이전트의 효율적인 제어를 위하여 조정 에이전트를 통해서만 응용 에이전트들이 다른 에이전트들과 교류할 수 있다. 응용 에이전트들은 자신들의 능력을 조정 에이전트에게 미리 선언해 놓음으로써, 자신이 잘 처리할 수 있는 작업의 수행을 요구 받을 수 있도록 한다.

OAA 구조에서는 클라이언트 개념에 해당하는 응용 에이전트들이 조정 에이전트가 갖고 있는 블랙보드에 해결할 작업을 기록하는 식으로 진행된다. 조정



(그림 1) OAA 시스템의 구조
(Fig. 1) OAA system architecture

에이전트나 응용 에이전트들은 모두 독립적인 하나의 프로세스이다. 에이전트들은 미리 정의된 통신 프로토콜에 의해 메세지를 주고 받을 수 있다. 따라서 각 에이전트는 네트워크가 연결된 컴퓨터라면 어디에나 놓여질 수 있고, 네트워크를 매체로 원거리에 있는 에이전트와 상호 협력하면서 작업을 처리할 수 있다.

OAA 시스템의 특성은 사용자의 작업지시를 위탁받아서 에이전트 간의 교류를 통하여 분산적으로 실행해 주며, 사용자가 원하는 에이전트를 만들어 OAA 기반구조에 연결만 하면 기존에 이미 만들어진 다른 기능 에이전트와 협력하며 사용자의 요구를 수행해 준다. 또한 OAA 시스템은 작업지시를 쉽게 하기 위해 사용자가 멀티모달 입력이 가능하다. 따라서 사용자는 원하는 작업을 음성과 펜을 사용하여 자연어로 지시하면, 에이전트 간에 상호 협력하여 원하는 작업을 수행해 준다.

3.1 조정 에이전트와 응용 에이전트

조정 에이전트는 일종의 글로벌 데이터를 저장관리하고, 어떤 에이전트가 어떤 일을 해결할 수 있는가를 알아내며, 분산적으로 사용자 요구를 처리할 때 응용 에이전트들 간의 정보교류를 스케듈링하고 관리한다. 따라서 조정 에이전트는 문제를 해결하는 동안 어느 에이전트에게 어떤 작업을 맡길 것인가를 판단하고, 적절한 응용 에이전트를 선택하여 작업을 요구하는 조정기능을 수행한다. 응용 에이전트들 간의 모든 교류는 에이전트 통신언어(ICL:Interagent Communication Language)을 이용하여 블랙보드를 중개자로 해서만 이루어진다. 조정 에이전트의 주된 임무는 ICL을 분석하여 이를 해결해 줄 수 있는 에이전트에게 전달해주는 것이다. 따라서 에이전트들 간의 교류는 미리 예측할 수 없는 방향으로 진행되며 이때 조정 에이전트는 일종의 중개자 역할을 한다.

응용 에이전트는 특정한 영역의 작업을 전문적으로 처리하는 역할을 한다. 하나의 응용 에이전트는 블랙보드와의 교류를 위한 통신 층과 지식베이스 층으로 구성된다. 지식베이스는 기존 프로그램의 데이터파일의 이용, 응용 프로그램의 API의 이용, 스크립트언어의 이용 또는 오퍼레이팅 시스템에서 발생되는 이벤트 등을 이용하여 기존 프로그램의 기능을 이용할 수도 있다. 지식베이스 아래에는 기존의 독립적인 응용 프로그램들, 예를 들면 mail, calendar 또는 database 프로그램 등이 붙을 수 있다.

각각의 에이전트들은 사용자나 다른 에이전트의 요구를 만족시켜주기 위하여 정보를 주거나 어떤 행위를 할 수 있다. 또한 어떤 조건이 만족되는가를 감시하기 위하여 트리거를 설치할 수 있다. 즉, 트리거는 비동기적인 에이전트의 수행을 지원한다.

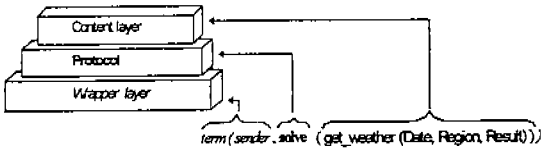
OAA 시스템을 수행시키면 응용 에이전트는 조정 에이전트에 연결한다. 연결한 후 응용 에이전트는 조정 에이전트에게 자신이 제공 가능한 기능을 알려준다. 조정 에이전트는 이러한 기능이 필요하면, 해당 에이전트에게 ICL을 사용하여 요구사항을 보낸다. 해당 에이전트는 요구사항을 분석한 후 처리하여 조정 에이전트에게 처리결과나 처리상태정보를 보낸다. 따라서 사용자나 각 에이전트는 OAA 시스템이 제공하는 다양한 기능을 사용할 수 있다. 예를 들면, 다른 응용 에이전트의 기능을 요구하거나 트리거를 설치하거나 조정 에이전트의 데이터 공유 영역에 데이터를 조작하는 등 다양한 기능을 에이전트 간에 서로 공유할 수 있다.

3.2 에이전트 통신언어(ICL : Interagent Communication Language)

분산된 네트워크 환경 하에서의 에이전트들은 서로 다른 플랫폼 위에서, 서로 다른 프로그래밍 언어를 이용하여, 서로 다른 사람들에 의해서, 서로 다른 시간대에, 서로 다른 문제 해결 방법을 적용하여 구현되는 경우가 많기 때문에 다양성을 갖는다. 따라서, 이들 에이전트들간의 인터페이스는 일정하지 않다. 이를 극복하고 에이전트들간의 협동작업을 수행하기 위해서는 에이전트 간의 통신언어가 필요하다. OAA 시스템에서 사용하는 에이전트 통신언어인 ICL은 에이전트들 상호간에 통신을 할 때 단일화(unification)와 백트래킹(back tracking)을 이용하기 때문에 프롤로그 프로그램 언어의 확장으로서 설계되었다.

ICL은 (그림 2)에 나타낸 것처럼 content 층, protocol 층, wrapper 층으로 구성된 3개의 층으로 이루어진다. Content 층은 에이전트 간에 교류되는 메시지 내용을 나타낸다. Protocol 층은 에이전트들 간의 약속된 규칙을 의미한다. 예를 들면, solve라는 프로토

콜이 담긴 메시지를 받은 에이전트는 solved라는 프로토콜이 담긴 메시지를 되돌려 주어야 한다는 규칙이 내포되어 있다. Wrapper 층은 에이전트가 그 내용을 보고 그 메시지가 ICL로 해석되어 질 수 있는 메시지인지 아닌지 판단하며, 어느 에이전트가 보낸 것인가를 알아낼 수 있도록 한다.

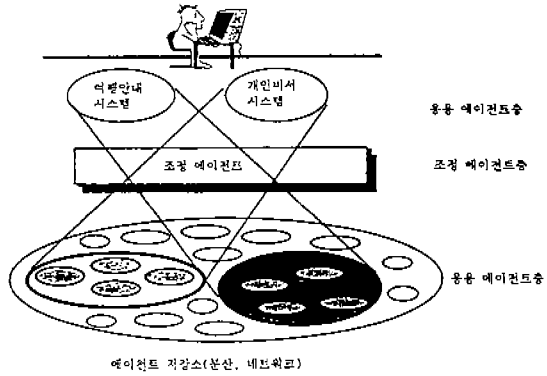


(그림 2) ICL의 구조
(Fig. 2) ICL architecture

OAA 시스템에 연결된 모든 에이전트는 ICL로 표현된 기능사양들을 정의하여 조정 에이전트에게 공지해야만 한다. 이것은 조정 에이전트에 의해 에이전트와 통신할 때 사용되고, 지시된 작업을 수행하기 위해 어느 에이전트에게 어떤 작업을 맡길 것인가를 판단하고 적절한 응용 에이전트를 선택하여 작업을 요구할 때 사용한다.

4. 에이전트 개발용 도구

OAA 시스템의 계층적 개념구조는 (그림 3)과 같이 응용 에이전트 층(child agent layer), 조정 에이전트 층(root agent layer), 에이전트 응용 층(agent application layer)으로 이루어져 있다. 응용 에이전트 층은 특정 서비스를 제공하기 위해 만들어진 응용 에이전트의 집합체로 개인의 전자우편을 관리해주는 전자우편 에이전트, 개인의 일정을 관리해주는 일정관리 에이전트, 인터넷에서 정보를 검색하는 웹 에이전트 등 아주 다양하다. 사용자는 필요에 따라 응용 에이전트를 만들어 추가하거나 삭제할 수 있다. 조정 에이전트 층은 응용 에이전트 층에 있는 응용 에이전트 중 특정 목적에 사용되는 응용 에이전트들을 관리하고 제어하는 조정 에이전트들의 집합체이다. 에이전트 응용 층은 특정 목적을 위해 만들어진 OAA 시스템의 집합체로 여행 관련 서비스를 제공하는 여행관리 시스템, 사용자의 비서의 역할을 하는 비서 시스템 등 응용 에이전트 층과 조정 에이전트 층에서 제공하는 에이전트들을 조합하여 만들 수 있다.

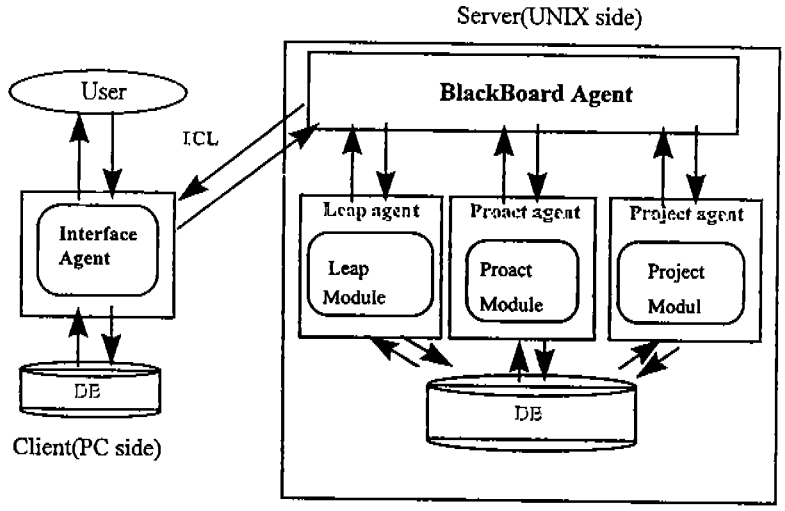


(그림 3) OAA 시스템 개념도
(Fig. 3) Conceptual model for OAA system

ADT는 앞에서 제안한 에이전트 개발도구의 요구사항인 확장성, 이식성, 관리성, 공유성, 조작성 등을 바탕으로 OAA 시스템 계층적 개념구조의 각 층을 쉽게 만들 수 있고 또한 이것을 통합 관리할 수 있도록 설계구현한다. 즉, ADT는 OAA 시스템을 사용하는 사용자가 OAA 시스템으로부터 새로운 서비스를 받기 위하여 새로운 에이전트를 생성하거나 특정 응용에 적합한 OAA 시스템을 구축할 때, OAA 시스템에 대하여 약간의 지식만 알고 있으면 원하는 에이전트를 쉽게 만들어 OAA 시스템에서 동작시키고, 이미 만들어진 에이전트들을 관리할 수 있다.

4.1 ADT의 구조 및 동작원리

ADT의 구조는 (그림 4)에 나타낸 것과 같이 OAA 구조를 그대로 갖는다. ADT의 구성은 하나의 조정 에이전트와 네 개의 응용 에이전트인 인터페이스 에이전트, 프로액트 에이전트, 리프 에이전트, 프로젝트 에이전트 등으로 이루어져있다. 인터페이스 에이전트는 개별 에이전트 개발에 관련된 입출력, 새로 만들어진 에이전트의 자연어 인터페이스 개발에 관련된 입출력, 특정 응용 영역에 적합한 에이전트 시스템 개발에 관련된 입출력 등 에이전트 시스템을 만드



(그림 4) ADT의 구조
(Fig. 4) ADT architecture

는데 필요한 데이터의 입출력을 전담한다. 프로액트 에이전트는 개별 에이전트 개발에 관련된 처리, 리프 에이전트는 새로 만들어진 에이전트의 자연어 인터페이스 개발에 관련된 처리, 프로젝트 에이전트는 특정 응용 영역에 적합한 에이전트 시스템 개발에 관련된 처리 등 사용자의 작업요구에 대한 실제적인 처리를 한다. 조정 에이전트는 인터페이스 에이전트와 다른 에이전트 간의 정보전달을 제어하며, 에이전트 간의 정보전달은 ICL을 이용한다.

ADT의 동작 원리는 다음과 같다. 먼저 사용자가 인터페이스 에이전트에서 제공하는 멀티모달을 이용하여 작업을 요구하면 이것을 분석하여 ICL로 변형한 후 조정 에이전트에게 전달한다. 이때 입력된 데이터는 클라이언트에 있는 DB에 저장한다. 조정 에이전트는 이것을 분석하여 해당 에이전트에게 전달하고 해당 에이전트는 실제적인 처리를 한다. 처리가 끝나면 처리결과를 ICL로 변형한 후 조정 에이전트에게 전달한다. 이때 처리된 데이터는 서버에 있는 DB에 저장한다. 조정 에이전트는 이것을 인터페이스 에이전트에게 전달하고 인터페이스 에이전트는 처리된 결과를 사용자에게 보여준다.

4.2 ADT의 구현

ADT 구현은 크게 서버쪽과 클라이언트쪽으로 나눌 수 있다. 서버쪽은 Sparc. 10(UNIX)을 개발환경으로 해서 C와 Prolog를 사용했고, 클라이언트쪽은 IBM 486 PC 상에서 Visual Basic을 사용하였다. 에이전트 간의 통신은 TCP/IP를 기반으로 한 ICL을 사용하였다. ADT의 구성은 개별 에이전트를 생성할 때 유용한 도구인 프로액트(Proact), 새로운 에이전트에게 자연어 인터페이스를 연결할 때 유용한 도구인 리프(Leap), 이미 만들어진 응용 에이전트를 이용하여 특정 응용에 적합한 OAA 시스템을 구축할 때 유용한 도구인 프로젝트(Project) 등으로 이루어져있다. 이하에서는 실제로 구현한 ADT의 주요 구성요소인 프로액트, 리프, 프로젝트를 상세히 기술한다.

(1) 프로액트(Programmer's Agent Construction Tool)

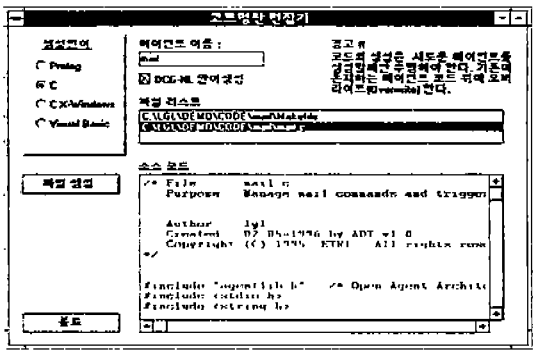
사용자가 응용 에이전트를 개발하려면 필요한 고유의 기능 외에 에이전트 기본기능, ICL 정의기능, 에이전트 등록기능 등을 만들어 주어야 한다. 프로액트는 이러한 기능을 쉽게 만들 수 있도록 지원하고 관리해 준다. 프로액트는 에이전트 등록기능, ICL 정의 기능, 공유문서 작성기능 및 열람기능, 에이전트 기본기능과 에이전트 틀 작성기능 등을 지원한다. 이러한 기능을 지원하기 위해 프로액트는 에이전트 등록기, ICL 편집기, 공유문서 편집기, 코드형판 편집기

(code template editor) 등을 가지고 있다.

에이전트 등록기는 새로운 에이전트를 만들 때 필요한 정보를 등록하기 위해 사용자에게 지원되는 편집기로서, 에이전트의 이름, 얼굴, 개발자, 버전, 타입, 설명문 등을 등록한다.

ICL 편집기는 에이전트의 기능을 정의하는 편집기로서, ICL 규칙 하에서 에이전트의 기능을 정의한다. ICL의 기능표현에 대한 일반적인 형식은 icl-id(⟨arg1⟩, ⟨arg2⟩, ..., ⟨arg N⟩)와 같이 기능구분자와 반복적인 여러 개의 인자들로 이루어져 있다. 여기서 arg X는 프롤로그 형식을 준한다.

공유문서 편집기는 에이전트의 기능을 사용자가 정의한 후 에이전트 관련 정보를 다른 사용자와 공유하기 위하여 자신이 만든 에이전트 관련 기능 및 참고사항을 기술하고, 멀리 떨어져 있는 사용자와 에이전트 관련된 정보를 일관된 문서형식으로 공유할 수 있도록 도와준다. 즉 공유문서 편집기는 웹서버브라우저가 읽을 수 있도록 HTML 형식으로 에이전트 설명문서를 생성하고 관리한다.



(그림 5) 코드형판 편집기
(Fig. 5) Code template editor

코드형판 편집기는 에이전트 이름, 목표 언어, 에이전트 기능 리스트가 주어지면 에이전트의 기능을 수행할 수 있는 프로그램의 기본 틀인 에이전트 코드형판을 자동적으로 생성해 준다. 이때 개발언어와 개발환경의 변화에 대한 이식성을 높이기 위하여 다양한 언어(Prolog, C, X-Windows용 C, Visual Basic)로 에이전트 코드형판을 생성할 수 있게 했다. (그림 5)는 전자우편 에이전트에 대하여 코드형판 편집기를 이용하여 코드를 자동적으로 만든 예이다.

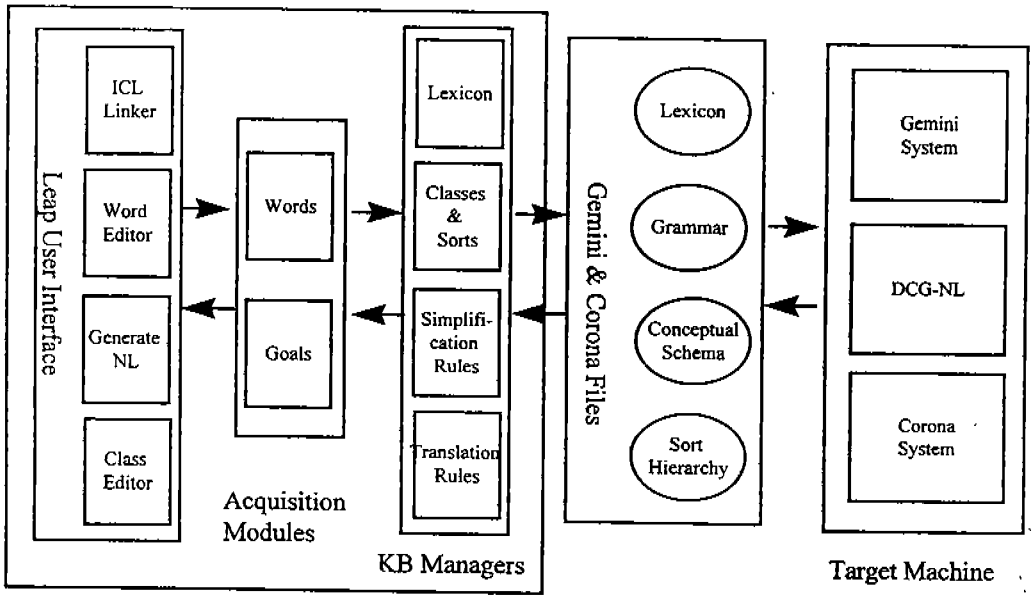
(2) 리프(Lexical Expertise Acquisition Platform)

에이전트 시스템의 자연어 인터페이스는 시간이 지나면서 에이전트가 새로 추가되거나 기존의 에이전트에 대해서 새로운 영역이 도입될 수 있다. 이때마다 에이전트 시스템의 자연어 인터페이스를 새로이 개발하는 것은 비효율적이다. 이러한 문제를 해결하기 위해 리프는 자연어 인터페이스 기본 시스템이 이용하는 언어 정보원(예:문법, 어휘사전, 개념, 개념의 계층구조, 프리디킷 및 인수)을 새로운 영역으로 대체 또는 추가 함으로써 자연어 인터페이스가 새로운 영역에 대해서도 동작할 수 있도록 했다.

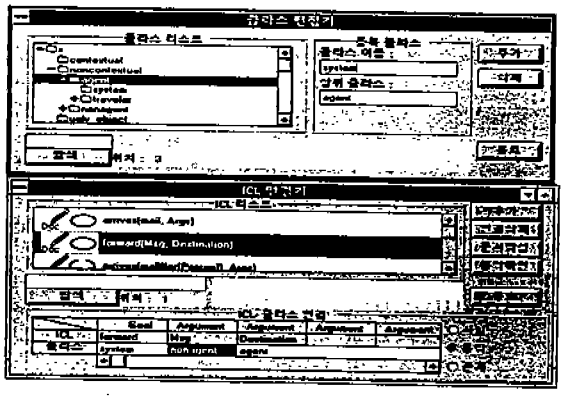
리프의 구조는 (그림 6)에 나타낸 것과 같이 사용자 인터페이스 모듈(leap user interface), 취득 모듈(acquisition module), 지식관리 모듈(KB managers)로 이루어져 있다. 개발자는 사용자 인터페이스에서 제공하는 클라스 편집기, ICL 연결기, 단어등록기, 자연어 생성기 등을 이용하여 자연어 처리에 필요한 새로운 정보를 입력한다.

클라스 편집기는 자연어 처리 시스템(GEMINI system[7])의 규칙엔진이 참조하는 클라스를 생성하고 관리한다. 클라스는 자연어 처리기에서 사용하는 한 개 이상의 단어를 하나로 추상화한 개념이다. 클라스 계층(class hierarchy)은 제미나이(GEMINI) 시스템이 인지할 수 있는 개념 카테고리로 트리구조를 가지며, 카테고리 간의 관계를 상위 클라스와 하부 클라스로 나타낸다. 상위의 클라스로 갈 수록 특정영역에 독립적이고 하위의 클라스로 갈 수록 특정영역에 종속적이다.

ICL 연결기는 ICL 편집기를 통하여 정의된 에이전트의 기능과 자연어에서 이용되는 단어를 연결해 준다. ICL과 자연어(클라스)를 연결할 때 ICL의 속성에 따라 사실(entity), 관계(relation), 동작(action) 중 하나의 형식을 설정해야 한다. 이것은 ICL 속성에 따라 처리되는 영역을 한정시켜 자연어 처리 에이전트가 ICL을 선택하기 위한 규칙선정의 처리효율을 높여준다. 사실은 ICL을 수행 시키면 클라스 오브젝트에 관한 정보를 엑세스 하고, 관계는 정보를 엑세스한다는 측면에서는 사실과 동일하나 이것은 클라스 간의 관



(그림 6) 리프의 블럭 구조
(Fig. 6) Architecture for the leap block



(그림 7) 클라스 편집기와 ICL 연결기
(Fig. 7) Class editor and ICL linker

계를 나타낸다. 동작은 ICL을 수행 시키면 어떤 행동이나 이벤트가 발생하여 실질적인 행동을 유발시킨다. (그림 7)은 클라스 편집기와 ICL 연결기를 이용하여 클라스와 ICL을 연결한 결과를 나타낸다.

단어 등록기는 자연어 처리 시스템(Gemini, DCG-NL[1, 11])과 음성인식 시스템(Corona system[8])에 입력될 문장 내의 단어와 어휘를 처리하기 위한 관련 정보(품사, 단어, 쓰이는 문장 형식)를 등록한다. 이때 간단한 일련의 질문과 예제문장을 제시하여 의미처리에 필요한 정보를 얻는다.

자연어 생성기는 자연어 처리 관련 지식을 가지고 새로운 에이전트에 필요한 언어정보원을 만들 때 목표 시스템을 설정한다. 현재, 자연어 생성기에서는 DCG-NL 자연어 파서, 제미나이 자연어 처리 시스템, 코로나 음성인식 시스템 등을 지원한다.

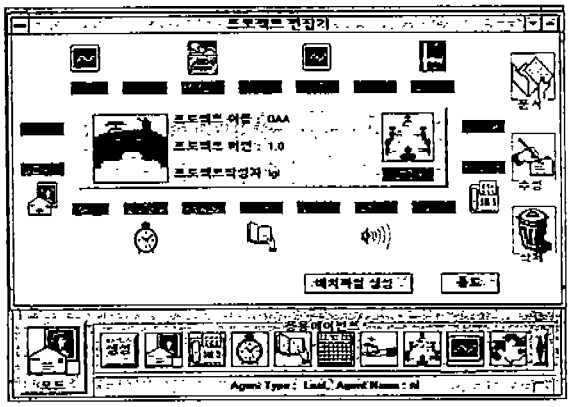
취득 모듈은 사용자 인터페이스로부터 입력된 정보를 단어정보(word information)와 목적정보(goal information)로 분리하여 관리하고 지식관리 모듈에서 필요한 정보를 확인한 후 부족한 정보에 대하여 개발자에게 요구한다. 이때 언어학에 대하여 별다른 지식 없이도 자연어 인터페이스 기본 시스템에 새로운 영역에 필요한 언어정보원을 수집하여 통합시킬 수 있도록 간단한 일련의 질문과 예제 문장을 제시하여 정보를 수집한다. 지식관리 모듈은 취득 모듈에서 수집한 정보를 이용하여 목표 시스템에서 사용되는 언어정보원을 만든다. 이렇게 해서 만들어진 언어정보원을 목표 시스템인 Gemini, DCG-NL, 코로나 시스템에 의해 사용된다.

(3) 프로젝트(Agent Configuration Manager)

프로액트를 이용하여 만들어진 새로운 응용 에이전트는 자동적으로 에이전트 저장소에 보관된다. 프로젝트는 에이전트 저장소에 있는 응용 에이전트를 조합하여 새로운 서비스를 제공하는 OAA 시스템을 만들 수 있다. 또한 이렇게 만들어진 시스템을 수행시키고 관리할 수 있다. 이러한 기능을 지원하기 위해 프로젝트는 프로젝트 등록기, 프로젝트 편집기, 프로젝트 파일 생성기, 프로젝트 수행기 등을 가지고 있다.

프로젝트 등록기는 새로운 에이전트 시스템을 만들 때 필요한 정보를 등록하기 위해 사용자에게 지원되는 편집기로서, 프로젝트의 이름, 얼굴, 개발자, 버전, 설명문 등을 등록한다.

프로젝트 편집기((그림 8) 참조)는 특정 응용에 적합한 응용 에이전트 시스템을 구축하기 위하여 필요한 응용 에이전트를 조합하여 하나의 시스템을 구성한다. 이때 사용자는 프로젝트 편집기를 이용하여 기존에 만들어진 응용 에이전트를 회의 테이블에 참가시킴으로써 응용 영역에 적합한 응용 에이전트 시스템을 만들 수 있다. 또한 참가한 에이전트의 특성 파라미터를 프로그램 수정 없이 조작함으로써 각 에이전트별 특성을 용이하게 저장할 수 있다.



(그림 8) 프로젝트 편집기
(Fig. 8) Project editor

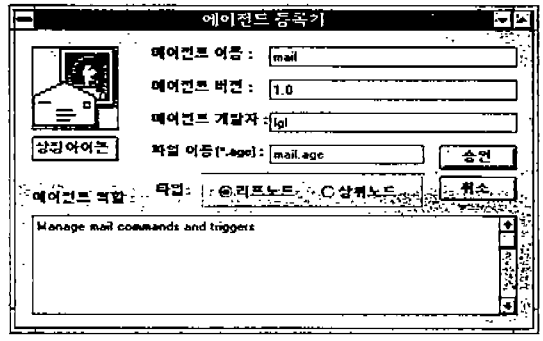
프로젝트 파일 생성기는 최종적으로 만들어진 응용 에이전트 시스템을 자동적으로 수행시키기 위하여 프로젝트 수행기에서 수행시킬 수 있는 스크립트 파일을 생성한 후 클라이언트쪽과 서버쪽의 데이터베이스에 저장한다.

프로젝트 수행기는 프로젝트 파일 생성기에 의해 만들어진 스크립트 파일을 수행시키고 그래픽하게 수행 중인 에이전트의 상태를 알려준다. 또한 수행되는 각 에이전트의 파라미터를 수행 중에 다이나믹하게 변경할 수 있고, 각각의 에이전트 프로세스를 관리해준다.

4.3 ADT를 이용한 에이전트 시스템 작성 및 실례

새로운 영역에 대한 OAA 시스템을 구축하려면 새로운 영역에 필요한 개별 에이전트를 만든 후, 에이전트 저장소에 저장된 응용 에이전트들을 이용하여 작성한다. 우선, 개별 에이전트를 만들기 위해서 에이전트 개발자는 다음의 7단계를 수행한다. 여기서는 실례로서 ADT를 사용하여 전자우편 에이전트의 작성 과정과 결과를 나타낸다.

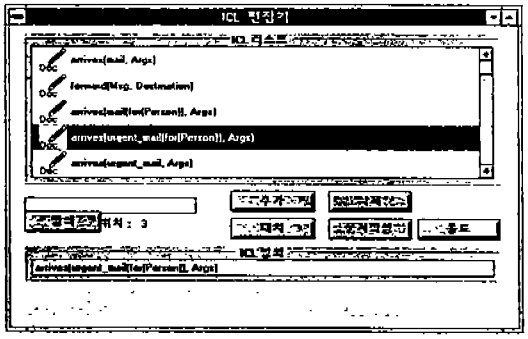
Step 1. 에이전트 정보등록: 에이전트 등록기를 이용하여 에이전트 생성에 필요한 정보 즉, 에이전트 이름, 얼굴, 만든 사람, 버전, 타입, 에이전트 설명문을 등록한다. (그림 9)는 전자우편 에이전트에 대하여 에이전트 등록기를 이용하여 작성한 결과이다.



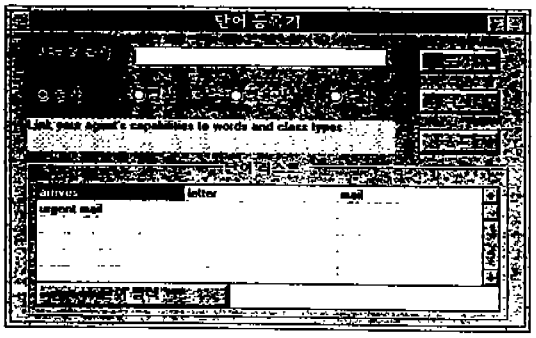
(그림 9) 전자우편 에이전트의 등록
(Fig. 9) Register informations for the email agent

Step 2. ICL 등록: ICL 편집기를 이용하여 에이전트의 기능들을 등록한다. 이때 ICL을 직접 입력

하는 직접방법과 기존에 제공되는 기능리스트를 호출한 후 유사한 기능의 ICL을 수정하는 수정방법이 있다. 수정방법은 조정 에이전트에 연결되어 현재 수행 중인 각 에이전트의 기능을 가져오는 방법과 특정 디렉토리에 있는 각 에이전트의 프로그램 소스로부터 가져오는 방법을 지원한다. 특히 ICL을 정의할 때 사용자는 자신의 로컬영역에서 제공되는 ICL 리스트 뿐만 아니라 다른 사용자가 이미 만들어 놓은 ICL을 웹서버브라우져를 이용하여 볼 수 있다. 따라서 새로운 ICL을 생성하기가 쉽고 많은 유용한 ICL을 직접 만들지않고서도 마치 자신이 만든 것처럼 이용할 수 있다. (그림 10)은 전자우편 에이전트에 대하여 ICL 편집기를 이용하여 작성한 결과이다.



(그림 10) 전자우편 에이전트의 ICL 등록
(Fig. 10) Register ICL for the email agent



(그림 11) 전자우편 에이전트의 단어 등록
(Fig. 11) Register words for the email agent

Step 3. 단어등록과 클라스 등록 : 단어 등록기를 이용하여 자연어 입력에 사용될 단어를 입력한다. 또한 클라스 편집기를 이용하여 새로운 영역에 대한 클라스를 등록한다. (그림 11)은 전자우편 에이전트에 대하여 단어 등록기를 이용하여 작성한 결과이다.

Step 4. 클라스와 ICL 연결 : ICL 연결기를 이용하여 ICL 형식(ICL type) 설정하고 ICL의 이름과 ICL 내의 각 아규먼트를 입력한다. 그런 후 ICL의 이름과 ICL 내의 각 아규먼트에 대응하는 클라스를 입력한다. 이것의 입력 방법은 클라스 편집기가 제공하는 클라스 계층에서 해당 클라스를 마우스로 선택하여 해당 위치에 끌어당겨 옮기기만 하면 된다. (그림 7)은 전자우편 에이전트에 대하여 클라스 편집기와 ICL 연결기를 이용하여 클라스와 ICL을 작성한 결과이다.

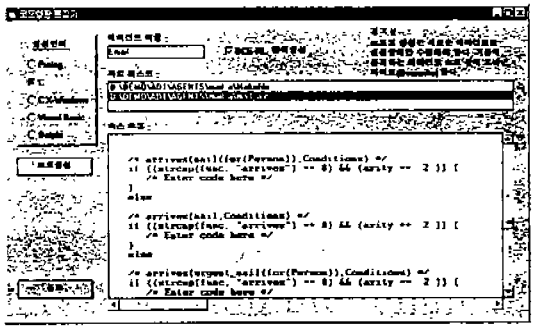
Step 5. 공유문서 작성 : 공유문서 편집기를 이용하여 에이전트의 기능 및 특성에 대하여 기술하고 이것을 웹서버브라우저가 읽을 수 있도록 HTML 형식으로 변환을 요구하면 공유문서 변환기는 사용자로부터 얻은 정보를 바탕으로 HTML 형식 파일을 생성한다. 사용자는 편집기를 이용하여 만들어진 HTML 파일을 수정할 수 있다. 이렇게 해서 만들어진 HTML 파일은 미리 정해진 특정 웹서버에 추가등록할 수 있고, 서버쪽의 데이터베이스에 저장이 되고 웹서버브라우져를 이용하여 열람할 수 있다.

(그림 12) 전자우편 에이전트의 공유문서 작성
(Fig. 12) Make the shared decumment for the email agent

(그림 9)는 전자우편 에이전트에 대하여 공유 문서 편집기를 이용하여 작성한 결과이다.

Step 6. 코드형판 생성:코드형판 편집기를 이용하여 원하는 언어(Prolog, C, X-Windows용 C, Visual Basic)를 선택하고 코드형판 생성을 요구하면 코드형판 생성기는 step 1부터 step 5까지의 과정을 통하여 얻은 정보를 바탕으로 코드형판을 생성한다. (그림 5)는 전자우편 에이전트에 대하여 코드형판 편집기를 이용하여 코드를 자동적으로 만든 결과이다.

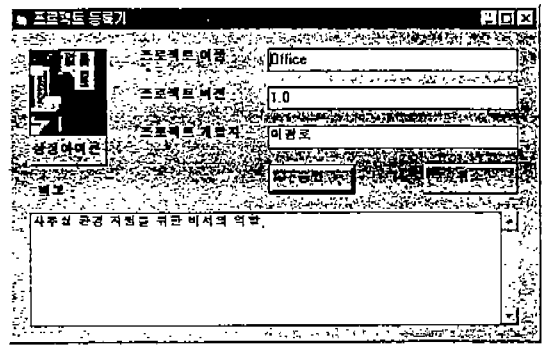
Step 7. 고유기능 작성:사용자는 코드형판 편집기를 이용하여 Step 6을 통해 생성된 코드형판 내에서 "Enter code here"라고 쓰여진 곳에 해당 ICL의 고유기능을 위한 코트를 작성한다((그림 13) 참조).



(그림 13) 전자우편 에이전트의 고유 기능작성
(Fig. 13) Make functions for the email agent

다음으로, 에이전트 시스템 개발자는 특정 영역에 적합한 새로운 에이전트 시스템을 만들기 위해서 다음의 3단계를 수행한다. 여기서는 실례로써 ADT를 사용하여 전자비서 시스템의 작성 과정과 결과를 나타낸다.

Step 1. 프로젝트 정보등록:프로젝트 등록기를 이용하여 프로젝트 생성에 필요한 정보 즉 프로젝트 이름, 얼굴, 만든 사람, 버젼, 타입, 프로젝트 설명문을 등록한다. (그림 14)는 전자비서 시스템에 대하여 프로젝트 등록기를 이용하여 작성한 결과이다.



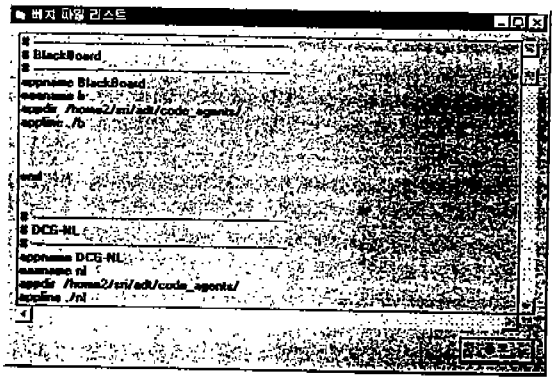
(그림 14) 전자비서 시스템의 등록
(Fig. 14) Register informations for the office secretary

Step 2. 응용 에이전트 시스템 편집:프로젝트 편집기를 이용하여 특정 응용에 적합한 응용 에이전트를 조합하여 하나의 시스템을 구성한다. 이때 사용자는 프로젝트 편집기 내에 있는 회의 테이블에 응용 에이전트를 참가시키기만 하면 응용 영역에 적합한 OAA 시스템을 만들 수 있다. 응용 에이전트를 참가시키는 방법은 에이전트 개발도구에서 관리하는 에이전트 저장소에 있는 에이전트를 마우스로 클릭한 후 회의 테이블 내의 원하는 위치에 옮겨 놓으면 된다. (그림 8)은 프로젝트 편집기를 이용하여 전자비서 시스템을 작성한 결과이다.

Step 3. 프로젝트 스크립트 파일 생성:프로젝트 편집기를 이용하여 만든 파일을 자동적으로 수행시키기 위하여 스크립트 파일 생성을 요구하면 프로젝트 파일 생성기는 프로젝트 편집기를 이용하여 얻은 정보를 바탕으로 프로젝트 스크립트 파일을 생성한다. 개발자는 편집기를 이용하여 만들어진 프로젝트 스크립트 파일을 수정할 수 있다. 이렇게 해서 만들어진 스크립트 파일은 프로젝트 수행기에 의해 수행된다. (그림 15)는 전비서 시스템에 대하여 프로젝트 편집기를 이용하여 스크립트 파일을 자동적으로 만든 결과이다.

5. 결 론

본 연구에서는 에이전트 시스템을 개발할 때 발생



(그림 15) 전자비서 시스템의 스크립트 파일
(Fig. 15) The script file for the office secretary

하는 문제점들을 해결하기 위해 에이전트 개발도구가 갖추어야 할 요구사항을 정리하고, 이러한 요구사항을 바탕으로 OAA 시스템을 목표 시스템으로 한 에이전트 개발도구를 개발했다. 앞으로 본 연구에서 개발한 에이전트 개발도구인 ADT를 사용하면 OAA 시스템에서 사용되는 에이전트를 용이하게 생성 및 기능확장이 용이하고, 새로 만든 에이전트에게 자연어 인터페이스를 용이하게 연결할 수 있고, OAA 시스템 관련된 자원을 효율적으로 관리할 수 있다. 따라서 에이전트 시스템 개발을 할 때 OAA 시스템에 대하여 전문가이든 비전문가이든 현재 사용하고 있는 많은 응용 소프트웨어를 손쉽게 에이전트화 하여 에이전트 시스템에 연결하여, 이미 제공되는 다른 에이전트의 기능과 조합한 고차원적인 서비스를 받을 수 있다. 그럼으로써 기존에 시스템 확장에 문제가 되는 인터페이스 부분에 대하여 개발자가 더 이상 신경을 쓸 필요 없이 자신이 필요한 부분에 대한 지식과 다른 응용 에이전트의 기능만 알고 있으면 쉽게 OAA 시스템의 기능을 확장할 수 있다.

앞으로의 연구 방향은 ADT를 이용한 새로운 에이전트 생성과 새로운 응용영역에 적합한 OAA 시스템을 구축, 인터넷 환경 지원을 위한 ADT의 기능확장, 이동(Mobile) 컴퓨팅 환경 지원을 위한 ADT의 기능확장, 안전한 분산환경을 위한 에이전트 간의 보안 기술개발 및 이를 지원하기 위한 ADT의 기능확장, 사용자의 애매한 질의를 처리할 수 있도록 에이전트 시스템의 지능화 기술개발 및 이를 지원하기 위한 ADT의 기능확장 등을 할 예정이다.

참 고 문 헌

[1] Cohen, A. Cheyer, M. Wang, and S.C. Baeg. An Open Agent Architecture. In O. Etzioni, editor, *Proceedings of the AAAI Spring Symposium Series on Software Agents*, pp. 1-8, Stanford California, March 1994. American Association for Artificial Intelligence.

[2] D.G. Schwartz. Cooperating heterogeneous systems:A blackboard-based meta approach. Technical Report 93-112, Center for Automation and Intelligent System Research, Case Western Reserve University, Cleveland, Ohio, April 1993. Unpublished Ph.D. thesis.

[3] Soon Cheol Baeg, Sang Kyu Park, Joong Min Choi, Myung Wuk Jang, and Young Hwan Lim, "Cooperation in Multi-agent Systems," Intelligent Computer Communications(ICC '95), Cluj-Napoca, Romania, pp. 1-12, June, 1995.

[4] 이광로, 박상규, "에이전트 기반 사용자 인터페이스 시스템:대표도우미," '94 가을정보과학회 학술발표논문집, 제 21권, 제 2호, pp. 667-670, 10월, 1994년.

[5] 백순철, 최중민, 장명욱, 박상규, 임영환, "이형 분산 환경에서 에이전트들간의 이형성을 극복하기 위한 멀티에이전트 기반구조," 정보과학회논문지(C), 제 2권, 제 1호, pp. 24-37, 3월, 1996년.

[6] 이광로, 박상규, 장명욱, 박치항, "에이전트 기반 사용자 인터페이스 개념 모델," '93 가을정보과학회 학술발표논문집, 제 20권, 제2호, pp. 1213-1216, 10월, 1993년.

[7] Dowding, J.M. Gawron, D. Appelt, J. Bear, L. Cherny, R.Moore and D. Moran. Gemini:A natural language system for spoken-language understanding. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 54-61, Columbus, Ohio, June 1993.

[8] Michael Cojen, Hy Murveit, Jared Bernstein, Patti Price, and Mitchel Weintraub. The DECIPHER

speech recognition system. In *IEEE ICASSP*, pp. 77-80, 1990.

[9] Tetsuo Kinoshita, "Agent Technologies: Recent Applications and Problems," 일본신학기보, AI95-40, pp. 41-47, November, 1995.

[10] 백순철, 최중민, 장명욱, 박상규, 민병의, 임영환, "플러그 앤드 플레이(Plug-and Play) 개념을 이용한 이형 응용 프로그램의 통합 기법," 정보처리 논문지, 제 2권, 제 6호, pp. 947-959, 1995년.

[11] Adam Cheyer and Luc Julia. Multimodal map: An Agent-based approach. In *Proc. of the International Conference on Cooperative Multimodal Communication(CMC/95)*, Eindhoven, The Netherlands, May 1995.

[12] Fumihiro Maruyama, "Agent and its application," 일본신학기보, KBSE95-34, pp. 25-26, November, 1995.

[13] Joongmin Choi, Sang-Kyu Park, Soon-Cheol Baeg, Myeong-Wuk Jang, Gowang-Lo Lee, and Young-Hwan Lim, "Message-Based Agent Communications in a Tightly Coupled Multiagent System," Fourth Golden West International Conference on Intelligent Systems(GWICS-95), San Francisco, CA., pp. 194-198, June, 1995.

[14] Barbara J. Grosz, Douglas E. Appelt, Paul Martin, and Fernando Perira, TEAM: An experiment in the design of transportable natural-language interfaces. Technical Note 356R, Artificial Intelligence Center, SRI International, Menlo Park, California, 1987.

[15] David L. Martin, Adam Cheyer, and Gowang-Lo Lee. Development Tools for the Open Agent Architecture. In *Proceedings of the* PAAM96, April 1996.

[16] Earl Craighill, Martin Fong, Keith Skinner, Ruth Lang, and Kathryn Gruenefeldt. SCOOT: An objectoriented toolkit for multimedia collaboration. In *Proc. of the ACM MULTIMEDIA 94 Conference*, pp. 41-49. San Francisco, CA, October 1994.

[17] 이광로, 박상규, 장명욱, 백순철, 민병의, 황승구, "OAA 시스템을 위한 에이전트 개발도구에 관한 연구," 한국정보처리학회 '96 춘계 학술발표논문집, 제 3권, 제 1호, pp. 227-230, 4월, 1996년.

이 광 로

1986년 일본 Fukuoka 공업대학교 전자기계과 졸업(학사)
1988년 일본 Ritsumeikan 대학교 전기공학과 졸업(석사)
1988년~현재 한국전자통신연구원 인공지능연구실 선임연구원
1994년~1995년 미국 SRI International(International Fellow)
관심분야: 에이전트 시스템, HCI, 패턴인식

박 상 규

1982년 서울대학교 컴퓨터공학과 졸업(학사)
1984년 한국과학기술원 전산학과 졸업(석사)
1984년~1987년 대림산업 전산실 근무
1989년~현재 한국과학기술원 전산학과 박사과정 수료
1987년~현재 한국전자통신연구원 인공지능연구실 선임연구원
관심분야: 에이전트 시스템, 정보 검색, HCI

장 명 욱

1990년 고려대학교 전산과학과 졸업(학사)
1992년 한국과학기술원 전산학과 졸업(석사)
1992년~현재 한국전자통신연구원 연구원
관심분야: 에이전트 시스템, 패턴인식

민 병 의

1982년 한양대학교 졸업(학사)
1984년 한국과학기술원 전기 및 전자공학과 졸업(석사)
1992년 한국과학기술원 전기 및 전자공학과 졸업(박사)
1984년~1987년 대림산업기술연구소

1987년~현재 한국전자통신연구원 인공지능연구실 실장
관심분야: 멀티미디어시스템, 에이전트

황 승 구

1979년 서울대학교 전기공학과 졸업(학사)
1981년 서울대학교 전기공학과 졸업(석사)
1986년 University of Florida 전기공학과(박사)
1982년 7월~현재 한국전자통신연구원 멀티미디어연구부 부장
1994년~1995년 미국 SRI International(International Fellow)
관심분야: 멀티미디어 시스템, HCI, 로보틱스