□ 기술해설 □

협력작업을 위한 에이전트 기반 소프트웨어

서울시립대학교 이재호

1. 협력작업의 개요

협력작업(collaborative computing)은 다수의 작업 참여자 간의 의사소통과 정보의 교환 및 공유가 원활이 이루어지도록 하여 공조활동(coordinated activities)을 돕는 컴퓨터 기술을 총칭한다. 특별히 그룹웨어(groupware)는 이러한 협력작업에 참여하고 있는 사람들을 지원하는 컴퓨터 소프트웨어로서 공유된 작업환경에 접근하는 인터페이스를 제공한다. 따라서 그룹웨어는 참여자 간의 정보 교환을 촉진하고 의사소통이 편리하도록 하여 협력과 공조를 돕는 것이 목적이다. 협력작업은 또한 컴퓨터지원 협동작업(Computer-Supported Coorperative Work, 약칭 CSCW)이라는 이름하에 연구되기도 한다. 컴퓨터지원 협동작업은 다시 전자메일을 통한 협력작업과 같이 한곳에서 생성한 결과를 나중에 다른 곳에서 이용하여 협력작업이 진행되는 비동기형(asynchronous) 작업과 화상회의와 같이 동시에 공동작업을 위한 협력이 필요한 동기형(synchronous) 작업으로 세분할 수 있다. 인터넷으로 연결된 고속의 개인용 컴퓨터의 보급과 함께 특히 동기형 협력작업에 대한 관심은 어느 때보다 높은 실정이며 더불어 이를 지원하는 소프트웨어를 효율적으로 개발하기 위한 기술에 대한 논의 또한 활발히 이루어지고 있다.

협력작업을 지원하기 위한 소프트웨어는 다수의 서로 다른 성격의 참여자의 공조작업을 돕는다는 면에서 다음과 같은 일반적 기능이 요구된다.

- 협력작업의 참가자는 원하는 작업의 상세한 절차(how)를 지정하기보다는 작업의 개요(what)만을 지정할 수 있어야 한다.
- 목적지향적(goal-directed)으로 작업을 수행하되 관련된 주변환경의 변화에 한정된 시간내에 반응하여야 한다.
- 하나 이상의 작업에 동시에 관여할 수 있어야 하며, 현재 진행중인 작업보다 우선 순위의 작업이 요구되거나 또는 긴급을 요하는 작업이 발생하면 현재 진행중인 작업을 일시 중지하여 높은 순위의 작업을 먼저 처리한 후 중지했던 작업으로 복귀할 수 있어야 한다.
- 협력작업 수행 중 발생하는 실패를 감지할 수 있고 실패에 대한 처리 대책을 갖고 있어서 실패로부터 복구할 수 있어야 한다. 실패의 발생과 복구 결과를 또한 사용자에게 알리거나 도움을 청할 수 있어야 한다.
- 트랜잭션(transaction)의 개념을 지원하여 일련의 여러 하위 작업을 한개의 상위 작업으로 묶어 처리할 수 있어야 한다.
- 기존의 응용 프로그램의 기능을 협력작업에 이용하기 위해서 협력작업 소프트웨어와 기존 프로그램을 쉽게 연계할 수 있어야 한다.
- 협력작업에 관련된 참가자 또는 대리 소프트웨어와 통신하는 수단을 갖고 있어야 하며 그들의 역할을 파악하고 있어야 한다.
- 신뢰할 수 없는 다른 참가자로부터 작업환경과 데이타를 보호할 수 있어야 한다.
- 신뢰할 수 있는 다른 참가자나 그 대리 소프트웨어가 요구한 원격 작업을 이해하고 실행할 수 있어야 한다.
- 지역적이나 원격적으로 발생하는 사건(event)이나 자료값의 변화를 감시하고 인식

하여 그에 따른 적절한 행동을 취할 수 있어야 한다.

- 지역적으로 발생한 사건에 관심을 가진 원격 참가자에게 발생 사건을 통지할 수 있어야 한다.
- 앞으로 발생할 가능성이 있는 공유자원에 대한 수요나 참가자 간의 갈등(conflict)을 예측하고 대비할 수 있어야 한다.

다음 절에서는 최근 각광받는 에이전트 기반 소프트웨어 개발기술과 위에 나열한 협력작업 지원 소프트웨어의 성격과의 연관성을 제시하기 위해 에이전트 시스템의 배경을 간략히 소개한다.

2. 에이전트 시스템의 개요

소프트웨어 에이전트(agent) 시스템은 전통적인 인공지능 시스템의 새로운 경향을 대변하는 중요한 개념이다. 에이전트 시스템에 필요한 기술은 기존의 클라이언트-서버(client-server) 모델을 초월한 혁신적 계산 모델을 제공할 것으로 기대되는 새로운 기술로서 미래의 분산형 소프트웨어 개발에 필수적인 핵심 기술이 될 것이다. 에이전트에 관한 연구는 지난 반세기 동안 인공지능 분야의 사실상의 주된 연구 과제였다고 해도 과언이 아니다. 인공지능 분야의 초창기인 50년대와 60대에 단순한 모델과 이론을 토대로 지능시스템을 구현하려던 낙관적 태도는 70년대와 80년대에 들어서 퇴색되면서 특정 기술 개발을 위한 구체적인 연구로 발전되었고 그 결과로 지식표현, 탐색기법, 게임 방법론, 전문가시스템, 기계학습, 자연어처리 등 인공지능 연구는 세분 또는 전문화되는 결과를 초래하게 되었다. 이러한 전문화 과정중 70년대 후반에 분산인공지능(Distributed Artificial Intelligence)이라는 새로운 연구 분야가 형성되어 다양한 종류의 분산인공지능 문제들을 에이전트라는 계산 단위와 에이전트 간의 상호작용을 토대로 해결하려는 시도가 시작되었다. 에이전트라는 용어는 80년대와 90년대에 걸쳐 데이타베이스나 운영체제 또는 전산망연구 등의 기존 컴퓨터 연구분야에서 사용되기 시작하였으며, 기본적으로는 컴퓨터 환경과 전산망 환경에 구애받지 않고 동일한 기능을 제공하는 소프트웨어 프로세스를 의미하였다. 90년대에 들어서 인터넷(internet)과 웹(web)의 폭발적 확장에 힘입어 웹 정보환경하의 소프트웨어 에이전트의 출현을 맞게 되었다. 이러한 에이전트는 컴퓨터 사용자를 대신하여 주어진 일을 수행하는 대행자나 중개자 역할을 한다. 최근에 들어서는 특히 개인비서(personal assistant) 역할의 에이전트를 개발하려는 연구도 활발히 진행되고 있다. 에이전트가 무엇인가를 정의하기 위한 수많은 논의에 버금가는 많은 정의가 있지만, 에이전트는 계속적으로 환경에서 지각된 것과 내부 지식을 바탕으로 추론하며 그 결과 행동하여 환경에 영향을 미치고 또한 다른 에이전트(컴퓨터 사용자도 포함)와 의사소통하는 지속적으로 존재하는 소프트웨어 요소라는 공통점을 갖고 있다 [3].

에이전트는 에이전트 구조(architecture)와 이에 바탕한 에이전트 프로그램으로 구현된다. 전형적인 에이전트 구조는 위에서 말한 공통적인 에이전트의 정의와 부합한다. 그림 1에 보인 바와 같이 에이전트 구조는 일반적으로 하부 구조로서 감지기(sensor)로부터 환경을 지각하는 지각(percetion)요소와 효과기(effector)를 통해 환경에 영향을 주는 행위(action) 요소를 가지고 있으며, 상위 구조로서 이 들을 연결하여 지각 요소로부터 입력을 받아 다음에 취할 행위를 결정하는 추론(reasoning)요소로 이루어진다.

다음 절에서는 이러한 에이전트의 일반 구조를 근간으로 개발된 UM-PRS[2] 에이전트 구조와 이들의 응용 예를 소개하고 더불어 앞서 나열한 협력작업 지원 소프트웨어의 성격과의

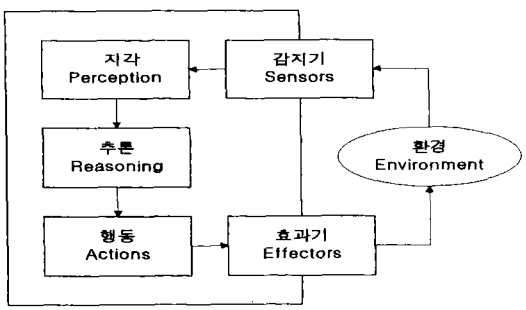


그림 1 에이전트의 일반 구조

연관성을 제시한다.

3. 협력작업 지원을 위한 UM-PRS 에이전트 구조

UM-PRS(University of Michigan Procedural Reasoning System)은 UM-PRS는 Georgeff 등에 의해서 초안된 PRS[4]를 C++를 써서 구현한 에이전트 구조로서 BDI 에이전트 구조[1, 5]에 근거하고 있다. UM-PRS는 감지기로 통해서 환경을 항상 감시하여 감지된 환경에 반응하여 플랜을 수행하는 반동적(reactive) 수행 특성을 가지고 있으며, 목적이나 데이타의 변화에 의해서 구동되어지는 목적구동(goal-directed) 및 데이타구동(data-directed)의 특성을 가지고 있다. 또한 에이전트의 기본행동이나 감지기능 등을 구현하는 기본함수들과 효율적으로 편리하게 연결시켜 사용할 수 있다. UM-PRS는 다음과 같은 실제 에이전트 시스템 구현에 중추적 역할을 담당했고 또한 활발히 이용중이다.

- ARPA가 지원하여 미국 미시간대학교에서 수행한 무인지상차량(UGV) 연구과제에서 로보틱차량을 자동제어하고 작전계획을 수립하는 로보틱 에이전트의 구현[6].
- 미시간대학교에서 진행중인 전자도서관(Digital Library) 연구과제, UMDL에서 중재(mediation) 및 촉진(facilitation) 에이전트의 구현[7].
- 인터넷상에서 다수의 이용자가 참가하여 모의 전투하는 Netrek 게임을 자율적으로 행하는 게임에이전트의 구현[8].
- 잠수함 및 전함의 승무원의 역할을 보조 또는 대체하기 위한 전함자동화(Ship Systems Automation)과제의 에이전트 시스템 구현[9].

이밖에도 UM-PRS는 정보전(information warfare) 에이전트 시스템, 전투제어 지능형 시스템 인터페이스(Combat Control Intelligent Systems Interface), Stealth Planning Associate(SPA)에 의한 전략정보시스템(Tactical Information Systems), Tactical Planning Associate(TPA)에 의한 방공대잠수함전(Offensive and Defensive Anti-Submarine Warfare) 등의 중추 기술로 사용되었다.

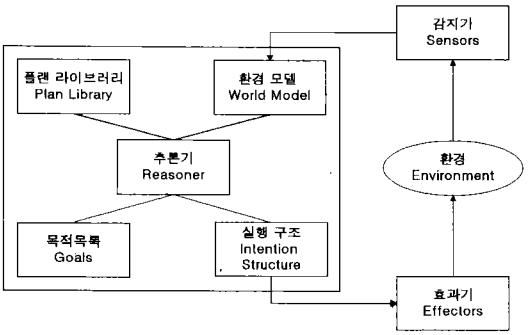


그림 2 UM-PRS 에이전트 구조

UM-PRS의 구조는 그림 2에 보인 바와 같이 앞서 그림 1의 에이전트의 일반 구조를 기본으로 한다. 이 그림에서 왼편의 큰 상자안에 있는 요소들이 에이전트의 추론 기관에 해당된다. 추론 기관은 다음의 요소들을 갖고 있다.

- 환경모델(world model)

환경모델은 관계식(relation)으로 표현된 사실(fact)들을 저장한 데이타베이스이다. 초기 사실들은 사용자가 미리 저장할 수도 있고 실행 중에 새로 첨가, 삭제, 또는 수정될 수도 있다.

- 목적(goal)목록

목적은 에이전트가 이루거나 유지하려는 상태로서 에이전트에게 주어진 임무를 나타낸다. 목적은 다시 에이전트의 상위(top-level)목적과 상위목적을 실현하기 위해 플랜을 수행중 시작되는 부수목적(subgoal)으로 분리된다. 목적은 ACHIEVE라는 단어 뒤에 목적의 이름을 써서 다음과 같이 지정한다.

ACHIEVE hole_dug ;

목적의 이름 뒤에 인수를 두어 매개변수를 갖는 목적을 지정할 수도 있다. 예를 들어 위의 목적, hole-dug에 hole의 폭과 깊이를 인수로 지정하려면 다음과 같이 할 수 있을 것이다.

ACHIEVE hole_dug 3 17 ;

목적의 중요도는 : PRIORITY라는 지정어를 사용하여 다음과 같이 지정한다.

ACHIEVE hole_dug : PRIORITY 5 ;

이 밖에도 지정한 목표를 특정한 플랜만을 써서 달성하려면 : BY라는 지정어 뒤에 원하는

플랜의 이름들을 나열할 수도 있다.

ACHIEVE hole_dug : BY dig_hole_with_shovel ;

유사하게 지정한 목표를 특정한 플랜을 제외한 다른 방법으로 달성하려면 : NOT-BY라는 지정어 뒤에 원하지 않는 플랜의 이름들을 나열할 수도 있다. 또한 : BY와 : NOT-BY 둘 다 동시에 지정할 수도 있다.

• 플랜 라이브러리(plan library)

플랜 라이브러리는 에이전트가 목적 성취를 위하여 이용할 수 있는 플랜들을 갖고 있다. UM-PRS의 플랜은 Knowledge Area, 또는 줄여서 KA라고 불리며 반드시 다음 두 항목을 지정해야 한다.

• PURPOSE : 플랜을 통해서 달성할 수 있는 목적을 나타낸다. 이는 앞서 설명한 목적(goal)과 대응된다.

• BODY : 지정된 업무를 달성하는데 필요한 단계적 수행절차(procedure)를 나타낸다. 수행절차는 기본행동(primitive action)과 부수목적(subgoal)들을 조건문이나 반복문으로 결합하여 나타낸다. 기본행동은 환경모델에 새로운 사실을 첨가하거나 삭제 또는 수정하는 연산을 포함한다.

이밖에도 UM-PRS의 플랜은 다음과 같은 추가적인 요소들을 지정할 수 있다.

• NAME : 플랜의 단순한 식별자이다.

• DOCUMENTATION : 플랜의 기능에 대한 설명문이다.

• CONTEXT : 플랜을 적용하기 위한 제약조건을 나타낸다.

• PRIORITY : 플랜의 우선순위를 지정하는 함수이다. 이는 앞에서 예로 보인 목적의 우선순위와는 별도로 플랜 자체의 우선순위를 지정할 수 있다.

• FAILURE : 플랜 수행에 실패한 경우 취할 단계적 수행절차를 나타낸다.

다음은 UM-PRS 간단한 플랜의 한 예이다. 이 플랜은 hole_dug이라는 목적을 실현하기 위한 플랜이며, 환경모델에 "has_shovel True"이라는 사실을 갖고 있을 때만 적용할 수 있다. 주어진 목적을 실현하는 방법은 세 가지 기본행동을 하는 수행절차로 이루어진다.

```
KA {
    NAME : dig hole with shovel
    PURPOSE : ACHIEVE hole_dug ;
    CONTEXT : FACT has_shovel True ;
    BODY :
        EXECUTE get_shovel ;
        EXECUTE dig_hole_with_shovel ;
        EXECUTE other_actions ;
    PRIORITY : 5 ;
}
```

• 실행구조(intention structure)

실행구조는 플랜을 실행하여 주어진 목적들을 실현하는 진전 상황을 기억하는 일종의 스택(stack)이다. 실행구조를 통하여 목적을 실현하는 과정을 중지하거나, 재개, 취소, 또는 계속할 수 있다.

• 추론기(Reasoner)

추론기는 판독기(interpreter)라고도 불리우며 전체 시스템의 수행을 관장한다. 추론기는 환경모델이나 목적목록에 변화가 생기면 (1) 현재 목적목록에 있는 목적들을 실현할 수 있는 플랜 중에서 현재의 상황에서 적용할 수 있는 플랜을 플랜의 PURPOSE와 CONTEXT로 부터 선별한다. (2) 선별된 플랜은 다시 목적의 우선순위와 플랜의 우선순위의 결합에 의해서 순위가 결정되고 가장 높은 순위를 갖는 플랜이 실행구조에 넣기위해 선택된다. (3) 이때 현재 실행구조가 비어있거나 새로 선택된 플랜이 우선순위가 실행구조에 있는 플랜의 우선순위보다 높으면 새로 선택된 플랜이 실행구조에 놓여진다. (4) 환경모델이나 목적목록에 변화가 없으면 실행구조에 놓인 플랜의 수행절차에 있는 아직 실행되지 않은 절차를 실행한다. 이 절차는 목적목록이나 환경모델을 변화시킬 수 있으며, 변화가 있으면 앞에 나열한 단계의 처음부터 다시 반복한다. 이러한 판독 방식은 새로운 상황에서 보다 중요한 목적이 생기면 그 목적을 실현하기 위해서 전환하는 것을 허용한다.

4. 협력작업의 성격과 UM-PRS 에이전트

앞절에서는 UM-PRS의 에이전트 구조와 구성 요소들을 간략히 살펴보았다. 이 절에서는 UM-PRS로 구현된 에이전트 시스템이 앞서 제기한 협력작업 지원 소프트웨어의 성격과의 연관성을 차례로 보여준다.

• 협력작업의 참가자는 원하는 작업의 상세한 절차(how)를 지정하기보다는 작업의 개요(what)만을 지정할 수 있어야 한다. UM-PRS를 포함한 에이전트 시스템에서 사용자가 에이전트에게 특정 작업을 지정하는 대표적인 방법은 에이전트에게 목적(goal)을 부여하는 방법이다. 목적은 작업의 상세한 절차가 아닌 바로 작업의 개요이다. 특히 UM-PRS에서는 상위목적을 부수목적으로 세분할 수 있어 작업의 개요를 사용자가 원하는 수준의 개요로 지정할 수 있는 융통성이 있다. 목적을 이루는 상세한 절차는 같은 목적에 대해서 하나 이상의 에이전트 플랜으로 작성되어 플랜 라이브러리에 저장하여 필요에 의해 보강하거나 교체할 수도 있다.

• 목적지향적(goal-directed)으로 작업을 수행하되 관련된 주변환경의 변화에 한정된 시간내에 반응하여야 한다. UM-PRS는 기본적으로 목적지향적이다. 다시 말해 목적목록에 있는 목적을 현재 환경에 적합한 최적의 플랜을 찾아 실현하는 기본 구조를 가지고 있다. 반면에 플랜을 실행하는 과정에서도 주변 환경의 변화를 끊없이 감시하고 변화된 환경에 반응하는 반동적 특성 또한 갖고 있다. 특히 한정된 시간내에 반응하는데 필요한 수행속도와 예측가능한 반응시간을 부여하기 위해서 Lisp 대신에 전부 C++로 구현되었다.

• 하나 이상의 작업에 동시에 관여할 수 있어야 하며, 현재 진행 중인 작업보다 우선 순위의 작업이 요구되거나 또는 긴급을 요하는 작업이 발생하면 현재 진행중인 작업을 일시 중지하여 높은 순위의 작업을 먼저 처리한 후 중지했던 작업으로 복귀할 수 있어야 한다. UM-PRS의 실행구조와 추론기는 다중프로세서 기계의 도움없이 하나 이상의 작업, 즉 목적에 관여할 수 있다. 작업 간에 긴급 정도는 각각의 목적에 주어진 우선순위와 플랜에 주어진 우선순위를 결합하여 나타낼 수 있으며 이에 따라 작업 간의 전환이 처리된다. 특히 UM-PRS의 우선순위는 상수값으로 주어지는 것이 아니고 함수로 지정될 수 있기 때문에 상황의 변화에 따른 우선순위의 조정도 손쉽게 이루어질 수 있다.

• 협력작업 수행 중 발생하는 실패를 감지할 수 있고 실패에 대한 처리 대책을 갖고 있어서 실패로부터 복구할 수 있어야 한다. 실패의 발생과 복구 결과를 또한 사용자에게 알리거나 도움을 청할 수 있어야 한다. UM-PRS에서는 플랜이 수행에 실패한 경우에 취할 수습절차를 각각 플랜의 FAILURE: 난에 지정하여 실패로부터 복구하는 방법을 지정할 수 있다. 수습절차는 UM-PRS가 제공하는 모든 기본행동과 조건 또는 반복문을 사용하고 더불어 환경모델의 정보를 이용하여 복잡한 수습절차를 표현할 수 있다.

• 트랜잭션(transaction)의 개념을 지원하여 일련의 여러 하위 작업을 한 개의 상위 작업으로 묶어 처리할 수 있어야 한다. 트랜잭션 개념을 지원하기 위해 UM-PRS는 ATOMIC이란 구성자를 제공하여 여러 개의 기본행동들을 한 개의 기본행동처럼 수행하는 기능을 제공한다. 이러한 ATOMIC 행동을 수행하는 동안에는 다른 작업으로 전환되는 것을 방지하여 트랜잭션의 개념을 지원한다.

• 기존의 응용 프로그램의 기능을 협력작업에 이용하기 위해서 협력작업 소프트웨어와 기존 프로그램을 쉽게 연계할 수 있어야 한다. UM-PRS은 C++로 구현되었으며 사용자의 새로운 응용프로그램에 필요한 기본행동들을 C++ 함수로 손쉽게 추가할 수 있다. 또한 UM-PRS 자체는 C++ 라이브러리 형태로 제공되기 때문에 다른 응용프로그램에 손쉽게 링크되어 에이전트 기능을 제공할 수 있다.

• 협력작업에 관련된 참가자 또는 대리 소프트웨어와 통신하는 수단을 갖고 있어야 하며 그 들의 역할을 파악하고 있어야 한다. UM-PRS는 기본행동의 일부로 다양한 종류의 통신수단을 제공한다. Unix의 소켓(socket), CMU에서 제공하는 TCX, 분산 시스템의 표준인 CORBA 등 다양한 통신 수단의 채용이 용이하다.

• 신뢰할 수 없는 다른 참가자로부터 작업 환경과 데이타를 보호할 수 있어야 한다. 이러한 성격은 비단 UM-PRS뿐만 아니라 에이전트를 기반으로하는 모든 소프트웨어가 제공할 수 있는 기능이라고 할 수 있다. 객체지향언어에서 객체단위로 데이타를 보호할 수 있는 것처럼 에이전트 기반 소프트웨어에서는 에이전트라는 명확한 보호 단위를 제공한다.

• 신뢰할 수 있는 다른 참가자나 그 대리 소프트웨어가 요구한 원격 작업을 이해하고 실행할 수 있어야 한다. 앞의 기능과 상호 보완적인 특성이라고 할 수 있으며 에이전트 기반 소프트웨어에 가장 적합한 기능이라고 볼 수 있다. 원격 작업은 에이전트 간에 목적을 서로 전달하여 다른 에이전트에게 작업을 요구할 수도 있고, 모빌(mobile) 에이전트의 경우 수행할 작업을 자체를 전달하거나 에이전트 자신이 직접 이동하여 작업을 수행하는 것도 가능하다. UM-PRS에 모빌에이전트의 이동성을 부여하기위해 UM-PRS의 모든 기능을 Java언어로 재 구현한 JAM 에이전트는 기본적인 모빌에이전트의 이동성을 갖고 있다.

• 지역적이나 원격적으로 발생하는 사건(event)이나 자료값의 변화를 감시하고 인식하여 그에 따른 적절한 행동을 취할 수 있어야 한다.

• 지역적으로 발생한 사건에 관심을 가진 원격 참가자에게 발생 사건을 통지할 수 있어야 한다.

• 앞으로 발생할 가능성이 있는 공유자원에 대한 수요나 참가자 간의 갈등(conflict)을 예측하고 대비할 수 있어야 한다.

위에 나열된 나머지 성격들은 사실상 한 에이전트의 구조에 의해서 지원될 수 있는 기능이라기 보다는 복수에이전트(multiagent) 기반에서 지원되어야할 공조(coordination)문제들이다. 에이전트 기반 소프트웨어에서 연구되고 제안된 방법들을 채용할 수 있다.

5. 결 론

이 글을 통하여 협력작업을 실현하는 소프트웨어의 성격과 기능을 규정하고, 새로이 각광받는 에이전트 기반 소프트웨어 개발기술과 협력작업 지원 소프트웨어와의 연관성을 제시했다. 지면 제약상 이 글에서는 UM-PRS의 가장 기초적인 기능만을 소개하였음을 밝힌다. 에이전트 개념을 소프트웨어 개발기법으로 본다면, UM-PRS에서 제공하는 것과 같은 에이전트 기능들은 협력작업 지원 소프트웨어와 같이 복잡하고 분산된 소프트웨어 개발에 필요한 핵심 소프트웨어 기술로 자리잡을 것으로 기대된다.

참고문헌

[1] Anand S. Rao and Michael P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems,* pp. 312-319, San Francisco, California, June 1995.

[2] Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Patrick G. Kenny. UM-PRS: an implementation of the procedural reasoning systems for multirobot applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS'94),* pp. 842-849, Houston, Texas, March 1994.

[3] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents,* chapter 1, pp. 1-23. Morgan Kaufmann, 1997.

[4] Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert,* 7(6) : 34-44, December 1992.

[5] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91),* pp. 473-484. Morgan Kaufmann, Cambridge,

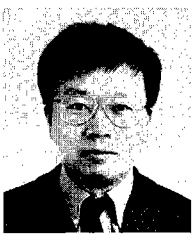
MA, April 1991.

[6] Patrick G. Kenny, Clint R. Bidlack, Karl C. Kluge, Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Terry Weymouth. Implementation of a reactive autonomous navigation system on an outdoor mobile robot. In *Association for Unmanned Vehicle Systems Annual National Symposium (AUVS '94)*, pp. 233-239, Detroit, MI, May 1994. Association for Unmanned Vehicle Systems.

[7] Jos M. Vidal and Edmund H. Durfee. Task planning agents in the UMDL. In *Proceeding of the 1995 Intelligent Information Agents Workshop*, 1995.

[8] Marcus J. Huber. *Plan-Based Plan Recognition Models for the Effective Coordination of Agents Through Observation.* PhD thesis, University of Michigan, Ann Arbor, Michigan, 1996.

[9] Edmund H. Durfee, Marcus Huber, Michael Kurnow, and Jaeho Lee. TAIPE : Tactical assistants for interaction planning and execution. In *Proceedings of the First International Conference on Autonomous Agents (Agents '97)*, pp. 443-450, Marina del Rey, California, February 1997.



이 재 호

1985 서울대학교 계산통계학과 학사
1987 서울대학교 계산학전공 석사
1996~1998 미국 ORINCON 회사 근무
1997 미국 미시간주립대학 전산학 박사
1998 서울시립대학교 전자전기공학부 교수
E-mail : jaeho@ee.uos.ac.kr