에이전트 지향의 소프트웨어 모델링 방법론
(Agent Oriented Software Modeling Methodology)

김 민 정 † 이 승 연 †† 박 근 하 †† 박 원 영 ††† 박 수 용 ††††
(Min Jeong Kim)(Seung Yun Lee)(Ken Ha Park)(Won Young Park)(Soo Yong Park)

요 약 에이전트에 대한 연구는 최근에 들어와 그 응용분야가 점차 확대되어 가고 있으며, 에이전트를 근간으로 하는 소프트웨어도 점점 복잡화, 대형화되고 있는 추세이다. 이에 따라 에이전트를 근간으로 하는 소프트웨어 개발 방식도 좀더 체계적이며 공학적인 접근이 요구되고 있다. 본 논문에서는 에이전트 기반의 소프트웨어를 효율적으로 모델링하기 위하여 에이전트 추출, 내부 에이전트 모델링과 외부 에이전트 모델링 기법을 제안한다. 에이전트 추출은 실세계에 존재하는 객체들로부터 에이전트를 추출하는 방법을 사용했고, 내부 에이전트 모델링에서는 에이전트의 내부를 목표, 자기정보, 계획, 기능으로 나누어 각각의 모델링 방법을 제시했으며, 외부 에이전트 모델링에서는 다중 에이전트 시스템에서 에이전트의 이동성과 에이전트간의 메시지 교환을 중심으로 한 모델링 방법을 제안한다.

Abstract As the concepts of agent are widely accepted and used in many application areas, agent oriented software is becoming large and complex. To support a systematic development of such software, agent oriented software development methodology is necessary. For modeling agent oriented software efficiently, agent elicitation method, Intra and Inter agent modeling methods are proposed. Agent elicitation shows how to extract agents from entities in the real world. The modeling methods of agent's characteristics - Goal, Belief, Plan and Capability - are proposed in Intra agent modeling and in Inter agent modeling, the methods of agent's mobility and communication in multi-agent systems are proposed.

1. 서 론

현재 소프트웨어의 모습은 점차 복잡하고 대형화되어 가고 있으며, 이를 지원하기 위한 소프트웨어 환경 역시 이질적이고 분산된 실시간 환경을 지원하도록 요구되고 있다. 이렇게 시스템이 대형화되고 복잡해짐에 따라, 미래의 소프트웨어 환경에 적합한 하나의 해결책으로 부각되고 있는 에이전트 분야는 초기 인공지능 분야에서 점차 그 응용 분야가 산업부문, 공공부문으로 확대되어가고 있으며, 특히 전자메일을 필터링하는 에이전트에서부터 항공 교통 관제(Air-traffic Control) 와 같이 크고 목적이 뚜렷한 복잡한 시스템에 이르기까지 다양하게 사용되고 있다[1]. 또한 전자상거래 응용프로그램들과 같이 분산되고 이질적인 환경에서, 에이전트에 관한 이론들은 널리 받아들여지고 있으며 여러 응용 프로그램의 개발에 적용되고 있다.

이러한 에이전트 지향의 소프트웨어들이 개발됨에 따라 소프트웨어 공학적인 측면에서 에이전트 지향의 소프트웨어 개발을 위한 분석적인 접근 방법이 점점 중요해지고 있으며, 체계적인 개발 방법론에 대한 연구가 절실히 필요하게 되었다.

현재 에이전트지향의 소프트웨어 공학 분야에서의 연구는 크게 영역 분석 및 모델링 기술, 에이전트 아키텍처 설계 기술, 에이전트 구현 툴 및 환경 개발 기술의 세 분야로 나눌 수 있다. 영역 분석 및 모델링 기술은 주어진 문제 영역에서 에이전트화 되어야 할 부분을 찾아 모델링 함으로서 에이전트의 기능, 변화 등을 기술하

· 본 연구는 2000년도 정보통신부 학술진흥사업 및 서강대학교 산업기술연구소의 지원에 의한 것임.
† 비 회 원 : 한국전자통신연구원 S/W공학연구부 연구원
minjkim@etri.re.kr
†† 학생회원 : 서강대학교 컴퓨터학과
sylee@selab.sogang.ac.kr
bluc@sclab.sogang.ac.kr
††† 비 회 원 : 서강대학교 컴퓨터학과
wypark@selab.sogang.ac.kr
†††† 정 회 원 : 서강대학교 컴퓨터학과 교수
sypark@ccs.sogang.ac.kr
논문접수 : 1999년 6월 28일
심사완료 : 2000년 8월 18일

고 검증하며 소프트웨어 공학의 재사용 개념을 활성화 시키는데 중요한 역할을 하는 기술이며, 에이전트 아키텍쳐 설계 기술은 정의된 에이전트를 구현하기 위한 최적의 구조를 개발하는 기술로 객체 지향 그룹과 컴포넌트 지향 그룹의 연구 결과를 기반으로 이를 에이전트 아키텍쳐 설계에 응용하는 분야이다. 에이전트 구현 툴 및 환경 개발 기술은 에이전트들을 구현하기 위한 언어와 에이전트들 사이의 통신언어와 그에 따른 규약들을 개발하는 기술로서 최근 들어 활발히 연구되고 있는 분야이다[2]. 에이전트 지향의 소프트웨어 공학에서는 위의 세 분야를 포괄적으로 관리함으로써 분산된 환경에서의 소프트웨어 개발과 시스템 관리를 효율적으로 지원할 수 있기 때문에 최근 들어 국제적으로 많은 관심을 모으고 있다.

따라서, 에이전트 지향의 소프트웨어 개발의 체계적인 개발 방법을 위하여 [그림 1]과 같은 모델을 생각해 볼 수 있다. 이러한 모델에서 초기 단계의 활동이라 할 수 있는 에이전트 발견, 도메인 모델링, 에이전트의 내부 및 외부 모델링 방법을 제안한다.

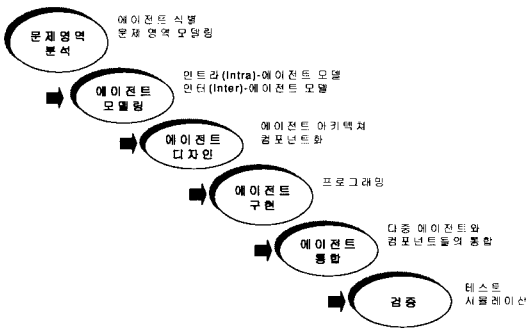

그림 1 에이전트 지향의 소프트웨어 개발 공정을 나타낸 폭포수 모델

본 연구는, 실세계는 에이전트와 객체가 공존하며, 에이전트는 능동 객체(Active Object)[1] 또는 분산 객체[3]의 모습이라는 개념 하에, [그림 1]의 공정 모델을 기반으로, UML(Unified Modeling Language)[4,5]을 이용한 문제 영역 분석 과정에서 얻어진 객체들에 앞으로 기술될 기준들을 적용시켜 에이전트화 시킬 객체들과 새로 생성해야할 에이전트들을 결정한다. 이렇게 생성된 에이전트 중심의 시스템을 구체적으로 모델링하기 위해, 본 논문에서는 크게 두 부분으로 나누어 모델링을 하였다. 즉 에이전트가 가지는 속성들과 행동들을 표현하기 위한 내부 에이전트 모델링(Intra agent modeling) 방법과, 여러 에이전트들 간의 메시지의 교환 및 이동성을 나타내기 위한 외부 에이전트 모델링(Inter agent modeling)을 제안한다.

본 논문의 3장에서는 에이전트 지향의 모델링 방법을 제시한다. 3.1절에서는 UML을 이용한 문제 영역 분석 방법을 제시하고, 3.2절에서는 에이전트 추출 규칙(agent selection rule)을 이용한 에이전트 추출과정과 제안된 에이전트-클래스 다이어그램(agent-class diagram)에 대해 기술한다. 3.3절에서는 목표(Goal), 자기정보(Belief), 계획(Plan), 기능(Capability) 모델링을 통한 내부 에이전트 모델링을 제안하며, 3.4절에서는 에이전트 이동 모델(Agent Mobile model)과 에이전트 통신 모델(Agent Communication Model)을 이용한 외부 에이전트 모델링을 제안한다. 마지막으로, 4장에서는 본 논문에서 제안한 에이전트 지향의 모델링 기법을 검증하기 위해 제한된 범위의 전자상거래 도메인 예제에 본 논문에서 제안한 모델을 적용시켰다.

2. 에이전트의 성질 및 기존의 에이전트 모델링 방법들

문제 영역 분석에 들어가기에 앞서, 에이전트는 어떠한 성질을 가지며, 에이전트와 객체와는 어떠한 차이점이 있는지를 살펴보고, 에이전트의 현재 개발 방향에 비추어 모델링 요소 및 방향을 살펴보고자 한다.

에이전트에 대한 개념은 1950년대 중반 John McCarthy로부터 시작하여 몇 년 후 Oliver G. Selfridge에 의해 정립되었다. 초기의 에이전트는 인공지능 분야에서 중점적으로 연구가 진행되었으나, 80년대 말부터 인공지능과 분리되어 독립적인 연구 주제로 대두되기 시작하였다.

이러한 에이전트의 성질은 Wooldridge와 Jennings[6]에 의하면 다음의 네 가지로 분류할 수 있다.

1. 자율성(autonomy) : 에이전트는 사람의 개입 없는 상태(환경)에 의해 독립적으로 자신의 행동을 결정한다.

2. 반응성(reactivity) : 에이전트는 어떠한 환경에 위치하여 환경을 지각하고 변화하는 환경에 반응할 수 있다.

3. 선행성(pro-activeness) : 환경에 단순히 반응하지 않고 목적 지향적인 행동을 한다.

4. 사회성(social ability) : 에이전트는 자신의 목표를 이루기 위해 다른 에이전트와도 상호 작용한다.

현재 에이전트에 관한 연구 방향은 크게 두가지 영역으로 나눌 수 있는데, 첫째는 위와 같은 에이전트의 성질

에 기반하여 에이전트의 자율적인 능력에 초점을 맞춘 인공지능적인 측면에서 자율적인 에이전트를 개발하고자 하는 노력이고[7], 둘째는 다중 에이전트 시스템의 분야로서 에이전트 통신 언어개발과 환경구축 및 에이전트간에 정보를 공유할 수 있는 방법에 관한 분야이다. 다중 에이전트 분야에 있어서는, 기존의 CORBA나 DCOM과 같은 분산환경에서 일어나는 문제점을 극복하고 보다 효율적인 분산환경을 제공하기 위한 이동 에이전트(Mobile Agent)가 연구되고 있다.

에이전트와 같이 합리적인 행동을 할 수 있는 복잡한 시스템을 어떻게 모델화 할 수 있을까 하는 연구는 계속되고 있으며, 그 중 비교적 성공적이라 할 수 있는 기존의 에이전트 모델로서는 Rao-Georgeff의 BDI(Belief - Desire - Intention) model[8]이 있으며, Kinney는 OMT를 확장시킨 MAS(Multi Agent System)를 위한 디자인 방법론을 제시하였다[9]. Kinney가 제시한 모델에서는 객체 지향적 모델링 기법에서의 장점인 추상화(Abstraction), 상속(Inheritance), 모듈화(Modularity)기법 등을 BDI 모델에 적용시켜 전체 시스템을 내부적인 측면과 외부적인 측면으로 모델링 하였다. Burmeister [10]는 OO(object oriented)기법을 확장시킨 에이전트 지향의 방법론을 제안하였는데 에이전트 지향 시스템을 에이전트 모델(agent model), 조직 모델(organizational model) 그리고 협동 모델(cooperational model)로 구분하여 나타내었다. 또한 최근 모빌 에이전트를 모델링 하고자하는 노력으로서 B. Falchuk와 A. Karmouch에 의한 Visual agent modeling[11]이 제시되었는데, 그들의 논문에서 에이전트들과 그들이 활동하는 환경에 존재하는 여러 요소들을 아이콘으로 정의한 뒤에 이아이콘들을 이용하여 에이전트의 이동성과 통신에 대한 모델링 방법을 제안하였다.

그러나, 시스템 분석단계에서 에이전트화 되어야 할 부분과 객체의 모습으로 존재해도 무관한 부분에 대한 명확한 구분이 미약하고, 다중 에이전트 환경 하에서의 에이전트들과 모빌 에이전트의 통합된 모델링 방법이 부족하므로 이를 보완하기 위하여 본 논문에서 새로운 에이전트 지향의 모델링 방법을 제안하고자 한다.

3. 에이전트 지향의 모델링 방법

실세계를 모델링하는 연구는 많이 진행되어 왔으나 본 논문에서는 에이전트를 중심으로 실세계를 모델링하려는 시도를 하였으며 이를 위하여 다음의 사실들을 발견하였다.

- 실세계(real world)는 에이전트와 객체들이 상호작용을 하면서 공존한다.
- 능동 객체(Active Object)를 에이전트라 할 수 있다[1].
- 에이전트는 비동기적으로 행동한다.
- 에이전트와의 상호작용은 메시지 교환에 의해 일어난다.

이러한 사실들을 기반으로, [그림2]에서와 같은 에이전트 지향의 모델링 과정을 제안한다. 모델링 단계는 [그림 2]에서 제시한 바에 따라, 크게 네 가지로 나누어 볼 수 있으며, 과정의 첫 번째인 문제 영역의 분석단계에서 UML 기반의 객체지향 분석 방법을 사용한다. UML은 이미 산업체에서도 이의 실용성이 입증된 모델링 언어로, 에이전트 추출의 초기단계에 객관성을 부여할 수 있다. 이를 통하여 시스템의 동적인 부분과 정적인 부분을 동시에 분석할 수 있어서 문제 영역을 깊게 이해할 수 있다. 문제영역 분석 과정이 끝난 후에는,

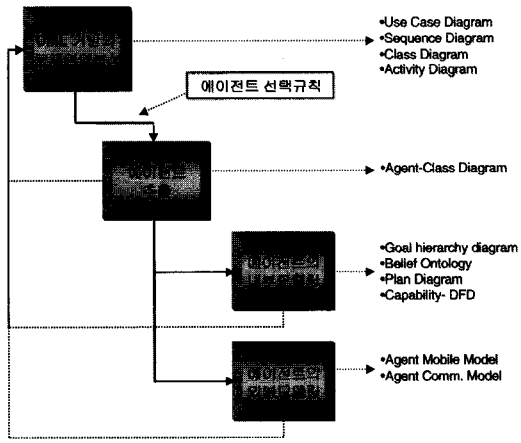

그림 2 에이전트 지향 방법론 공정

[표 1]에서 제시될 <에이전트 추출 규칙>을 적용하여 에이전트화 되어야 할 객체들과 새로 만들어야할 에이전트들을 결정하고, 이 과정에서 생성된 에이전트-클래스 다이어그램을 이용하여 에이전트와 클래스를 표현해 준다. 또한, 각 에이전트마다 내부속성을 크게 목표(Goal), 자기정보(Belief), 계획(Plan), 기능(Capability)으로 나누어 각각의 모델링 방법을 제안하고, 에이전트의 외부속성을 나타내기 위해서는 모빌 에이전트 모델과 에이전트 커뮤니케이션 모델을 이용하여 에이전트간의 관계를 모델링 할 것이다.

3.1 UML을 이용한 문제 영역 분석

본 논문에서는 문제 영역의 분석을 위해 UML[4,5]을 이용하여 문제 영역을 분석한다. 이를 통하여 시스템의 정적인 면과 동적인 면을 분석할 수 있으며, 에이전트화 할 수 있는 객체 및 새로 생성할 에이전트를 결정할 수 있도록 객체 지향의 분석 방법을 선행한다. 여기에 사용되는 각 다이어그램들은 다음과 같다.

1. 유스케이스 다이어그램(Use Case diagram) : 유스케이스 다이어그램은 유스케이스(Use Case)와 액터(Actor)(특별한 종류의 클래스)들의 집합과 그들간의 관계를 보여준다. 유스케이스 다이어그램은 시스템의 정적인 관점을 묘사하므로, 시스템의 행동들을 구조화하고 모델링 하는 데 중요하다.

2. 시퀀스 다이어그램(Sequence diagram) : 시퀀스 다이어그램은 메시지의 시간적인 순서에 중점을 둔 인터액션 다이어그램(Interaction diagram)이다. 객체들의 집합과 객체들에 의해 보내지고 받는 메시지들을 보여주며, 객체들은 클래스의 인스턴스들로서 이름이 있거나 없을 수도 있으며, 다른 것들의 인스턴스를 나타낼 수도 있다. 시스템의 동적인 관점을 묘사하기 위해 사용된다.

3. 클래스 다이어그램(Class diagram) : 클래스 다이어그램은 클래스와 인터페이스의 집합, 그들간의 협력과 관계들의 집합을 보여준다. 클래스 다이어그램은 객체지향 시스템의 모델링에서 가장 흔히 적용되는 다이어그램으로서, 시스템의 정적인 디자인 관점을 표현하기 위해 사용된다.

4. 액티비티 다이어그램(Activity diagram) : 시스템 내의 액티비티(Activity)와 액티비티들간의 흐름을 보여준여주는 액티비티들과, 행동하고 행동을 받는 객체들에다. 액티비티 사이의 순차적이거나 분기적인 흐름을 보의해 보여진다. 시스템의 동적인 관점을 묘사하기 위해 사용되며, 특히 시스템의 기능을 모델링 하는데 있어서 특히 중요하다. 또한, 액티비티 다이어그램은 객체들간의 컨트롤의 흐름을 강조한다.

3.2 에이전트 추출

UML에 의해 분석한 시스템의 동적인 면과 정적인 면을 고려하여, 클래스 다이어그램에 다음의 기준을 적용하여 객체들 중에 에이전트로 만들어야 하는 것들을 선택한다. 각 기준들은 앞에서 말한 에이전트의 기본 성질과도 밀접한 연관성을 갖는다.

에이전트의 기본 속성으로부터 생성된 에이전트 추출 기준에 의해 만들어질 수 있는 에이전트의 아키텍처를 Nwana가 제안한 에이전트 분류를 참고로 메타 규칙을 적용하여 4가지 그룹으로 제안한다. 각각의 분류는 스마트 에이전트(smart agent), 협력 학습 에이전트(collaborative learning agent), 협력 에이전트(collaborative agent), 인터페이스 에이전트(interface agent)이다.

표 1 에이전트 추출 기준

1. 자율성(Autonomy)
1.1 내부지식이 필요한가?
1.2 스스로 의사결정을 내리는가?
1.3 잘못된 입력, 예상치 못한 상황에 대응가능한가?
2. 적응성(Adaptation)
2.1 지속적으로 보완이 필요한가?
2.2 외부 객체와 상호작용이 있는가?
3. 협동성(Cooperation)
3.1 협력하여 작동하는가?
3.2 다중 쓰레드로 작동 가능한가?

표 2 메타 규칙

1. 에이전트가 자율성, 적응성 그리고 협동성을 갖는다면 그 에이전트를 "스마트 에이전트"라 한다.
2. 에이전트가 적응성, 협동성을 갖는다면 그 에이전트를 "협력 학습 에이전트"라 한다.
3. 에이전트가 자율성, 협동성을 갖는다면 "협력 에이전 트"라 한다.

이러한 기준에 의해 추출된 에이전트는 이동성이 필요한 지의 여부에 따라 이동 에이전트와 에이전트로 구분하며, 여기서의 에이전트는 이동성이 필요없는 에이전트, 즉 상주 에이전트를 의미한다.

이렇게 결정된 에이전트와 새로 생성되어야 할 에이전트 그리고 에이전트가 되지 못한 클래스들을 클래스 다이어그램의 발전된 형태인 에이전트-클래스 다이어그램으로 표현할 수 있다.

3.2.1 에이전트-클래스 다이어그램(Agent-Class Diagram)

본 논문에서 제안한 에이전트-클래스 다이어그램은 실세계에 객체와 에이전트가 공존한다는 사실에 근거하여, 시스템을 구성하는 에이전트들과 클래스들의 협력 및 그들의 관계를 나타낸다. 기존의 클래스 다이어그램에서 추가된 부분은 다음과 같다

개체(entity)

- Ⓐ : 에이전트를 나타냄 (이동성을 갖고 있지 않으며, 도메인 내에서 상주하는 에이전트)
- Ⓜ : 이동 에이전트를 나타냄.

• 클래스(Class) : 기존의 클래스를 나타냄

표기(notation)

• 협동(Cooperation)(—○—) : 에이전트와 에이전트간의 관계를 나타냄. 에이전트와 에이전트간의 협력이 필요함을 나타낸다.

• 고용(Employ)(E) : 에이전트와 클래스 사이의 관계를 나타냄. 에이전트가 기존의 클래스를 그대로 가져다 사용함을 나타냄.

각 에이전트의 내부에 목표, 자기정보, 계획, 기능 등의 내용을 정의할 수 있으며, 이들은 내부 에이전트 모델링에서 다시 자세하게 표현된다.

3.3 내부 에이전트 모델링(Intra Agent Modeling)

본 논문에서 제시하고자 하는 에이전트는 근본적으로 BDI 모델[8]에 기반을 두고 있으며, Shoham의 Mental Model[12]을 확장시킨 Thomas의 Reticular Agent Mental Models[13]을 참고로 하여 에이전트의 내부를 표현하고자 한다. 즉 에이전트의 내부를 크게 목표, 자기정보, 계획, 기능의 네 부분으로 보았으며, 각 부분에 대한 모델링은 다음과 같다.

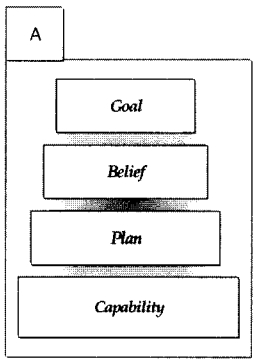

그림 3 에이전트의 내부 구성

3.3.1 목표 모델(Goal Model)

목표는 에이전트가 궁극적으로 달성하고자 하는 상태로서 KAOS[14]에서 사용된 Goal의 분류를 사용하였다. 각 목표는 문제 영역 분석 과정에서 유추되는 것으로서 Pattern_of_Goal [Objective]로 나타내며, Pattern_of_Goal은 크게 획득(Achieve), 중지(Cease), 유지(Maintain), 회피(Avoid), 최적(Optimize)의 다섯 종류로 나누어진다. 목표 계층 다이어그램(Goal Hierarchy diagram)에서는 KAOS[14]에서 나타낸 목표 구조(Goal structure)를 그대로 사용하여 각 목표에 대응되는 에이전트들간의 협력 관계를 나타내고자 하였다. 목표 계층 다이어그램에서 사용된 기호들은 다음과 같다.

개체

• 비 기능적 목표 : 전체 시스템에 의해 달성되어야하는 목표

• 기능적 목표 : 에이전트에 의해 달성되어야하는 기능적 목표

표기

• AND(⌒) : 목표를 이루기 위해 두 개이상의 하위 목표들이 동시에 만족되어야 함을 나타냄.

• OR(⌒) : 목표을 이루기 위해 OR 관계의 목표 중 하나만 만족되어도 됨을 나타냄.

• 상충(Conflict)(——) : 목표가 다른 목표와 목표하는 바가 상충될 경우를 나타냄.

각 목표는 계층구조를 두어 여러 개의 하위 목표로 나누어질 수 있으며, 각 목표와 하위 목표들 사이의 관계는 위에서 정의한 기호들로 나타낼 수 있다. 목표가 에이전트와 매핑될 수 있으며, 매핑되는 에이전트간의 관계는 목표 계층 다이어그램에서의 관계와 일치하여, 에이전트가 어떤 에이전트와 협력해야 하는지를 명확하게 알 수 있다.

3.3.2 자기정보 모델(Belief Model)

자기정보는 에이전트가 알고 있는 정보로서, 환경에 대한 정보 및 에이전트 자신에 대한 정보 등을 포함하며 계속적으로 갱신될 수 있다. 즉 자기정보는 주어진 작업을 수행하는 데 필요한 모든 지식 정보를 나타내며, 이러한 자기정보 중에서 여러 에이전트들 간에 체계적인 지식 구조로 만들어져서 공유되어야 하는 것, 또는 체계적인 구조로 관리해야할 필요가 있는 것들에 대해서는 온톨로지(Ontology)를 구축하여 나타낸다. 또한 이러한 온톨로지들은 Ontolingua[15]나 KIF(Knowledge Interchange Format)[16] 또는 XML 형식에 기반하여 구축될 수 있다.

예를 들어 에이전트가 기본적으로 가지고 있어야 하는 속성은 크게 이름과 주소(모빌 에이전트의 경우)이다. 그 외에 에이전트가 온톨로지로 가져야 하는 것은 UML 기반의 문제영역 분석과정에서 나타난 다이어그램들에서 유추될 수 있다. 이러한 온톨로지는 다시 크게 초기 단계에서 모든 구조가 결정되어질 수 있는 온톨로지와, 유입되는 정보에 따라 지속적인 변화가 필요한 온톨로지의 두 가지로 분류할 수 있다. 각 에이전트에 어떠한 온톨로지가 필요한지 결정하는 기준은 다음과 같다.

1. *초기단계에서 결정될 수 있는 온톨로지*
- 지속적인 변화가 필요 없이 초기에 전체 구조를 결정할 수 있다.
- 에이전트간의 통신을 위한 프로토콜이나 ACL(Agent Communication Language)등에 관련된 온톨로지 등은 초기에 온톨로지로 구성한다.

2. *지속적인 변화가 필요한 온톨로지*
- 클래스 다이어그램에서 속성 또는 오퍼레이션에 관련된 것들은 온톨로지의 가능성이 있다.
- 시퀀스 다이어그램에서 교환되는 메시지에 해당하는 것들은 온톨로지의 가능성이 있다.
- 에이전트가 가지는 정보 중에서, 체계적으로 지식화 되어야할 필요가 있는 것을 온톨로지로 구축한다.

각각의 온톨로지는 SHOE에서 제안한 표현방법에 따라 XML을 이용하여 나타낼 수 있다. 각 온톨로지마다 아이디(ID)를 결정하고, 온톨로지를 구성하는 범주(Category)를 구분하고, 이들간의 관계를 정의한다. 이렇게 결정된 요소들은 XML을 통하여 명확히 표현될 수 있다.

3.3.3 계획 모델(Plan Model)

계획은 에이전트가 목표를 달성하기 위해 취하는 행동들을 나타낸 것으로, 기존의 문제 영역 분석에서 나타난 시스템의 동적인 부분 분석의 지식에 바탕을 두고, 분석된 에이전트를 기반으로 시간의 흐름에 따른 에이전트의 행동변화에 중점을 두어 여기서 발생하는 에이전트간의 메시지 교환을 보여준다. 에이전트는 자신의 목표와 자기정보를 참조하여 행동을 결정할 수 있게 되는데, 이는 UML의 인터액션 다이어그램의 표현 방식을 확장한 에이전트 계획 시퀀스 다이어그램(*Agent plan sequence diagram*)을 통해 나타내어진다. 에이전트가 목표를 가지고 이동하는 모습(*mobile(Goal[Objective])*), 자기정보에 따라 온톨로지를 수정하는 상태(*update(Belief[Type_of_ontologies])*), 그리고 에이전트 간에 전달되는 메시지(*msg[message_context]*)나 클래스 이용(*employ*)등의 내용을 보여줌으로써, 계획 모델로부터 에이전트의 필요한 기능들을 뽑아낼 수 있다.

그림 4 에이전트 기능의 일반적인 형태

3.3.4 기능 모델(Capability Model)

기능은 에이전트가 수행할 수 있는 기능들로서 에이전트의 입력과 출력 그리고 입력에 따른 내부 상태의 변화에 대해서 기능 위주로 기술한다. 이는 DFD(Data Flow Diagram)을 이용하여 나타낼 수 있다.

3.4 외부 에이전트 모델링(Inter Agent Modeling)

외부 에이전트 모델링에서는 다중 에이전트 시스템 내에서 에이전트간의 메시지 교환과 에이전트의 이동성을 표현한다. 내부 에이전트 모델은 에이전트 통신 모델과 에이전트 이동 모델 의 두 가지 모델로 구성된다. 여기서 에이전트이동 모델은 에이전트가 이동할 때 고려되어야 하는 사항을 표현하며, 에이전트 통신 모델에서는 다중 에이전트간의 통신을 나타낸다. 이 두 가지 에이전트 모델은 서로 독립적으로 모델링 되는 것이 아니라 병렬적으로 이뤄진다.

3.4.1 에이전트 이동 모델(Agent Mobile Model)

에이전트 이동 모델에서는 이동 에이전트가 작업을 수행하기 위해 이동하는 모습과 목적지를 결정하는 메커니즘을 보여준다. 즉, 에이전트의 '이동성'이 어떤 기준으로 이뤄지는지, 이동 에이전트가 어떤 호스트로 이동하는지 등을 보여준다.

에이전트가 이동할 때에 외부 관점만으로는 어떻게 목적지와 경로를 결정하는 지 알 수 없다. 따라서 비록 '이동'이라는 외부적 측면의 내용이라도 에이전트의 내부적인 측면에 대한 표현이 필요하다.

3.4.2 에이전트 통신 모델(Agent Communication Model)

에이전트 통신 모델에서는 에이전트 사이의 통신에 대해 보여준다. 문제 영역에 존재하는 에이전트들 중에서 어떤 에이전트들 사이에 통신이 이루어지며, 그들이 주고받는 메시지의 형태는 어떤 것인지에 대한 정보를 보여준다. 또한 에이전트간에 주고받게 될 메시지에는 어떤 정보가 담겨져 있으며, 그러한 메시지는 어떤 과정을 통해 생성되는지를 보여준다.

4. 전자상거래에서의 예제

본 절에서는 앞서 제시한 실세계에서의 에이전트 추출과 모델링 기법을 전자상거래(Electronic Commerce : EC)의 예에 적용시켜보도록 한다. 적용하는 예의 문제영역과 시나리오를 소개하고 이를 앞서 제시한 기법들에 근거하여 에이전트 추출과 에이전트의 내부 및 외부 모델링을 적용시켜 보겠다.

4.1 문제영역과 시나리오에 대한 기술

이 예에서 적용하는 문제영역은 전자상거래 영역을

축소한 형태로서 Win98, WinNT, Solaris의 각기 다른 운영체제를 구축하고 있는 4개의 서버로 한정하였다. 즉, Win98 2개, WinNT 1개, Solaris 1개의 서버에는 모니터, 마우스, 스피커, 키보드의 컴퓨터 부품에 대한 모델명과 가격정보가 저장되어 있고, 에이전트는 각 서버들의 DB를 돌아다니는 이동 에이전트로서 각각의 DB에서 최적의 가격을 가진 상품을 구입하도록 돕는다. [그림 5]는 이에 대한 전체적인 모습을 추상화시켜 보여주며 적용 예에 대한 자세한 시나리오는 [표 3]과 같다.

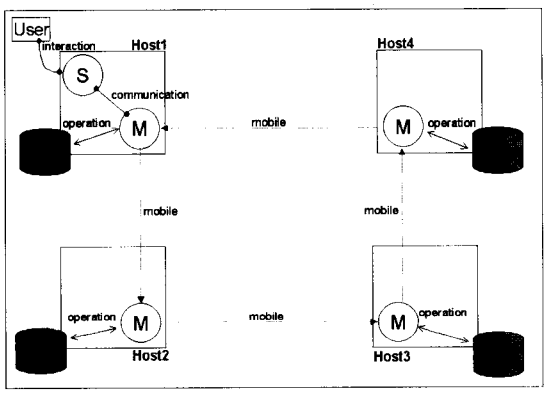

그림 5 문제영역에 대한 시스템 다이어그램

표 3 전자상거래 적용 시나리오

적용 시나리오

구매자는 인터넷을 통하여 컴퓨터 부품을 구입하려고 한다. 구매자는 상품 구매 에이전트에게 구입하고자 하는 컴퓨터 부품명과 사양을 입력한다. 상품 구매 에이전트는 분산되어 있는 서버들을 이동하면서 각 서버의 DB에 저장되어있는 상품들의 정보를 수집하고, 그 정보들을 비교하여 구입하려는 상품 중에서 최적의 가격을 가진 서버의 이름과 상품명, 상품 가격을 구매자에게 제시한다. 구매자는 에이전트가 제시한 정보를 보고 그 상품을 구입할 지의 여부를 결정한다. 구매자가 구입을 결정하면 구매 에이전트는 상품 구매에 대한 주문을 보낸다.

4.2 UML 기반의 에이전트 추출

전자상거래 예의 시나리오에서 UML기반의 분석을 이용하여 생성한 유스케이스 다이어그램, 시퀀스 다이어그램, 클래스 다이어그램, 액티비티 다이어그램에 앞서 제시한 추출기준을 이용하여 에이전트화 될 수 있는 객체들을 찾아 *에이전트 클래스 다이어그램*을 그린다.

4.2.1 유스케이스 다이어그램

본 예제의 시나리오에서는 구매자, 상품을 만들어내는 회사, 그 회사의 데이타베이스, 사용자인증을 얻기 위한 은행의 4개 액터가 존재하고, 고객과 질의를 주고받는 디스플레이(Display)기능, 고객이 원하는 상품을 찾는 검색(Search)기능, 고객의 주문을 처리해주는 주문(Order)기능으로 나눌 수 있고 이에 대한 유스케이스 다이어그램은 [그림 6]과 같다.

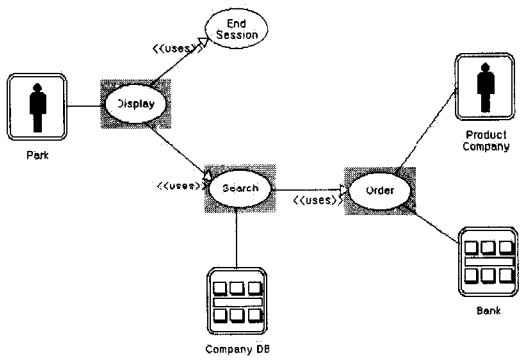

그림 6 EC예에 적용한 유스케이스 다이어그램

4.2.2 시퀀스 다이어그램

위의 유스케이스 다이어그램에서 각각의 유스케이스마다 적용되는 시퀀스 다이어그램을 만들 수 있는데 여기서는 이 예제의 가장 중심이 되는 Search 기능에 대한 시퀀스 다이어그램을 보이도록한다.

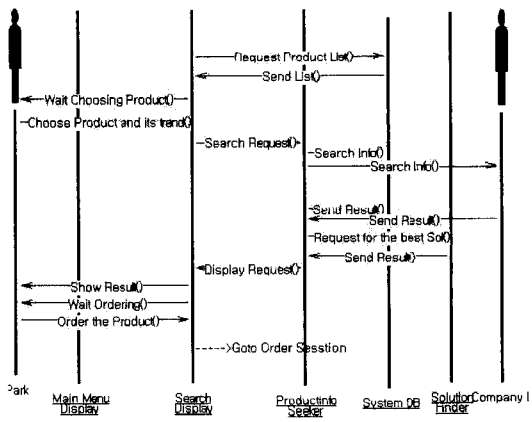

그림 7 EC예에 적용한 시퀀스 다이어그램

4.2.3 클래스 다이어그램과 액티비티 다이어그램

유스케이스 다이어그램과 시퀀스 다이어그램을 이용하여 객체로 만들어질 수 있는 부분에 대해 그의 속성과 기능을 정적으로 표현해주는 클래스 다이어그램과 동적으로 표현해주는 액티비티 다이어그램을 그려보면 [그림 8]과 같다.

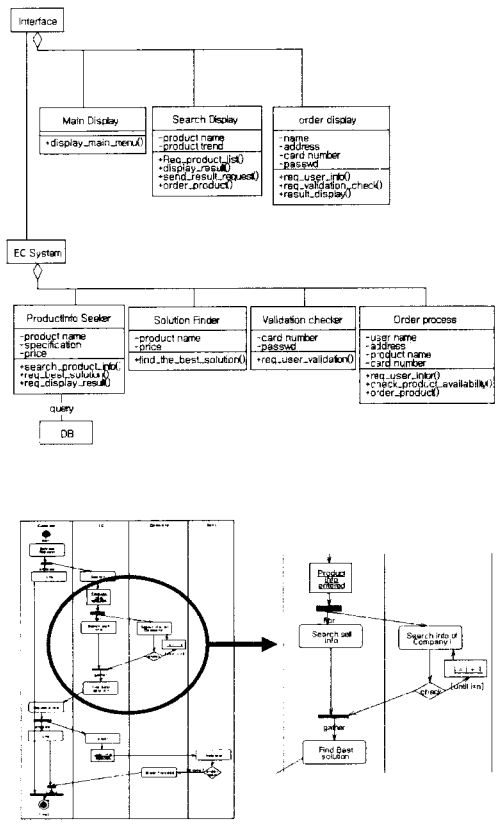

그림 8 EC에에 적용한 클래스 다이어그램(상)과 액티비티 다이어그램(하)

4.3 에이전트 추출과 에이전트-클래스 다이어그램

지금까지 그려온 UML 기반의 다이어그램들에 에이전트 추출 기준을 적용하여 에이전트화 될 수 있는 부분을 추출하도록 한다. 에이전트 추출기준을 적용하여 하나 이상의 클래스가 에이전트에 도달할 때 이를 그룹화하여 에이전트로 나타낸다. 위의 [그림 8]에서 제시한 클래스 다이어그램과 액티비티 다이어그램에서 3개의 에이전트(인터페이스 에이전트, 제품 정보 탐색 에이전트, 솔루션 파인더 에이전트)를 도출해냈으며 하나의 에이전트(중재 에이전트)를 지금까지 제시한 다이어그램들과 시나리오에 적용하여 보다 원활한 작업의 수행을 위하여 생성하였다. 각각의 에이전트에 대해 적용된 기준을 보면 [표 4]와 같다.

표 4 에이전트 추출 규칙의 적용 및 추출된 에이전트

인터페이스 에이전트(Interface Agent) : 질의를 주고받음으로써 사용자의 의도를 파악하고 그에 맞는 결정을 내려 제품 정보 탐색 에이전트(Product Info Seeker Agent)에게 메시지를 전달하고, 정보를 받아 원하는 결과를 사용자에게 보여준다.(규칙 1.1, 1.3, 2.2 적용)

제품 정보 탐색 에이전트(Product Info Seeker Agent) : 적용되는 예제에서 가장 핵심적인 역할을 담당하고 있는 에이전트로서 여러 데이타베이스를 돌아다니면서 정보를 찾는 기능, 사용자 인증기능, 주문처리기능을 가지고 있다. 이것은 여러 곳을 돌아다니면서 일을 수행해야 하는 분산환경에 적용되며, 다른 에이전트들과 협력하여 자신이 처리해야 하는 일을 수행하며 자신이 찾은 정보를 지속적으로 갱신시킬 수 있어야 한다.(규칙 1.1, 3.1, 3..2) 적용)

중재 에이전트(Coordinator Agent) : 외부 환경을 감지하여 이에 대한 변화가 일어날 때마다 자신의 내부 지식을 갱신시켜 제품 정보 탐색 에이전트가 돌아다닐 경로에 대한 지식을 제공한다. 이것은 예제에서 새로 생성된 에이전트로서 액티비티 다이어그램에서 표현된 알고리즘을 원활히 처리하기 위해 추가되었다. (규칙 1.2, 2.1, 3.1 적용)

솔루션 파인터 에이전트(Solution Finder Agent) : 제품 정보 탐색 에이전트가 찾아온 정보를 가지고 사용자가 원하는 최적의 solution을 결정한다. (규칙 1.1, 1.2, 3.2 적용)

[그림 9]는 [표 4]에서 나타난 에이전트들에 관한 *에이전트-클래스 다이어그램*이다. 여기서 이동의 특성이 있는 *제품 정보 탐색 에이전트*는 M 으로 나타내었다.

4.4 에이전트 내부 모델

에이전트의 내부 모델링은 목표, 자기정보, 계획, 기능의 속성을 표현하는 것이다. 앞서 제시한 모델링 기법들을 이용하여 에이전트의 각각의 속성에 대해 표현해 보도록 한다.

4.4.1 목표 모델링

모든 에이전트는 목표를 가지고 그것을 성취하기 위하여 여러 행동을 취한다. 본 페이퍼에서 적용하는 예에 나타난 4개의 에이전트들은 각각 목표를 가지고 있으며 그것을 나타내면 [표 5]와 같다.

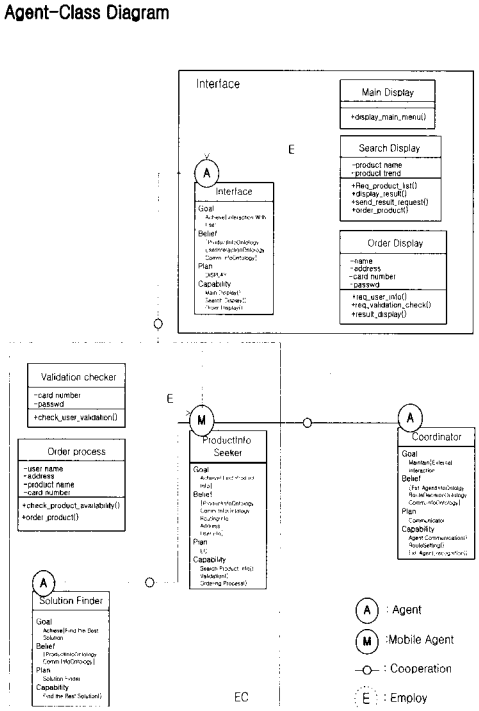

그림 9 EC예에서의 에이전트-클래스 다이어그램

표 5 각 에이전트의 목표 모델링

인터페이스 에이전트: Achieve[Interaction With User] 사용자와의 상호작용이 효율적으로 이루어지도록 하여 성공적인 결과를 이끌어낸다. **제품 정보 탐색 에이전트**: Achieve[Find Product Info] 사용자가 원하는 상품의 정보를 성공적으로 찾아낸다 **솔루션 파인더 에이전트**: Achieve[Find the Best Solution] 제품 정보 탐색 에이전트가 찾아온 상품정보에서 최적의 결과를 성공적으로 찾아낸다 **중재 에이전트** : Maintain[External Interaction] 외부환경의 변화를 지속적으로 감지하고 그에 대한 정보를 유지한다

각각의 에이전트가 자신의 목표를 성취하기 위해서는 그것의 하위 목표(Sub-goals)가 우선적으로 성취되어야 한다. [그림 10]은 제시된 예제에서 성취되어야 할 최종의 목표와 그의 하위 목표들 사이의 관계를 계층도로 표현하였다.

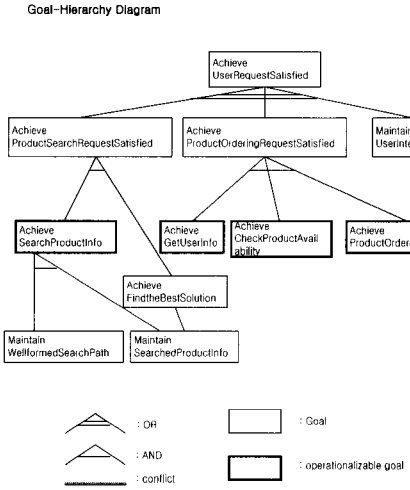

그림 10 EC 예에서의 목표 계층

4.4.2 자기정보 모델링

에이전트의 속성 중 자기정보는 주어진 작업을 수행하는데 필요한 모든 지식 정보를 말한다. 앞서 제시한 온톨로지 결정방법에 따라 전자상거래의 예에서 구축된 온톨로지는 [표 6]과 같다.

표 6 EC 예에서 온톨로지 및 각 온톨로지의 결정 기준

ProductInfoOntology : 사용자가 원하는 상품을 선택하기 위한 전반적인 지식. 상품의 종류와 그에 대한 사양에 대한 정보를 온톨로지로 구축한다.(클래스 다이어그램에서 검색 디스플레이(Search Display) 클래스의 속성 적용) **UserInteractionOntology** : 주문을 하기 위해 사용자가 입력하는 내용들에 대한 이해를 위해 이름, 주소, 신용카드등에 대한 전반적인 지식을 온톨로지로 구축한다.(클래스 다이어그램에서 주문 프로세스 (Order Process) 클래스의 속성 적용) **CommunicationOntology** : 에이전트들 사이의 협동을 이용할 때 메시지 전달을 위해 표준언어나 protocol과 같은 통신 지식을 온톨로지로 구축한다. (클래스 다이어그램에서 여러 클래스들의 요구(Request)기능 적용) **ExternalAgentInfoOntology** : 외부 환경을 감지하기 위하여 필요한 전반적인 지식을 온톨로지로 구축한다. (시퀀스 다이어그램에서 표현된 외부 액터(Actor)들과의 메시지 표현 적용) **RouteDecisionOntology** : 외부 회사의 데이타베이스를 돌아다니는데 필요한 라우팅 경로에 대한 지식을 온톨로지로 구축한다.(시퀀스 다이어그램에서의 외부 회사와의 메시지 전달 표현 적용)

위의 온톨로지중에서 제품 정보 탐색 에이전트는 ProductInfoOntology, CommunicationOntology를 가지며 그밖에도 RoutingInfo, UserInfo, 자신의 위치 등을 자기 정보로 가지고 있다.

여기서 각각의 온톨로지는 SHOE에서 적용한 것과 같이 XML을 이용하여 표현할 수 있는데 사용자와의 상호작용을 위해 필요한 UserInteractionOntology를 XML을 이용하여 표현하면 다음과 같다.

표 7 XML에 의한 UserInteractionOntology의 표현

```
<ONTOLOGY ID="UserInteractionOntology" VERSION="1.0">
<DEF-CATEGORY NAME="Actor"ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Card"ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Company"ISA="Actor">
<DEF-CATEGORY NAME="User"ISA="Actor">
<DEF-CATEGORY NAME="UserName" ISA="User">
<DEF-CATEGORY NAME="UserAddr" ISA="User">
<DEF-CATEGORY NAME="Master" ISA="Card">
<DEF-CATEGORY NAME="Visa" ISA="Card">
<DEF-CATEGORY NAME="Express" ISA="Card">
<DEF-CATEGORY NAME="CardNumber" ISA="Master, Visa,
Express">
<DEF-CATEGORY NAME="IssueDate" ISA="Master, Visa,
Express">
<DEF-CATEGORY NAME="ExpDate" ISA="Master, Visa,
Express">
<DEF-RELATION NAME="Issue">
<DEF-ARG POS="1" TYPE="Company">
<DEF-ARG POS="2" TYPE="Card">
</DEF-RELATION>
<DEF-RELATION NAME="Hold">
<DEF-ARG POS="1" TYPE="User">
<DEF-ARG POS="2" TYPE="Card">
</DEF-RELATION>
```

4.4.3 계획 모델링

에이전트의 속성 중 계획은 목표를 수행하기 위해 해야 할 작업에 대한 진행계획을 말한다. 이것은 UML의 인터액션 다이어그램의 표현방식을 따라 에이전트들 사이의 메시지 교환을 명시함으로써 작업의 진행순서를 표현했다. [그림 11]은 제품 정보 탐색 에이전트의 계획 속성을 나타낸 것이다.

4.4.4 기능 모델링

에이전트의 속성 중 기능은 작업을 수행하기 위해 필요한 기능들을 말한다. 이는 DFD로 나타낼 수 있는데

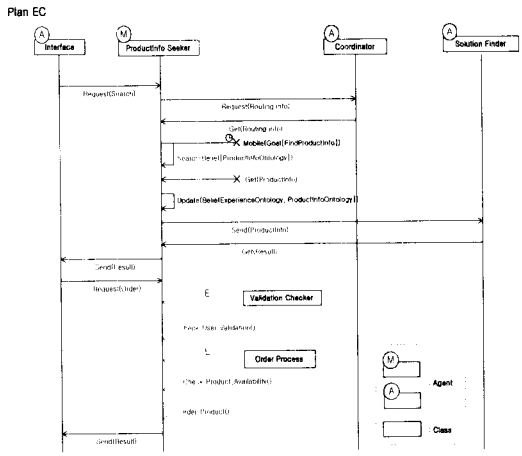

그림 11 EC예의 제품 정보 탐색 에이전트에 대한 계획 시퀀스 다이어그램

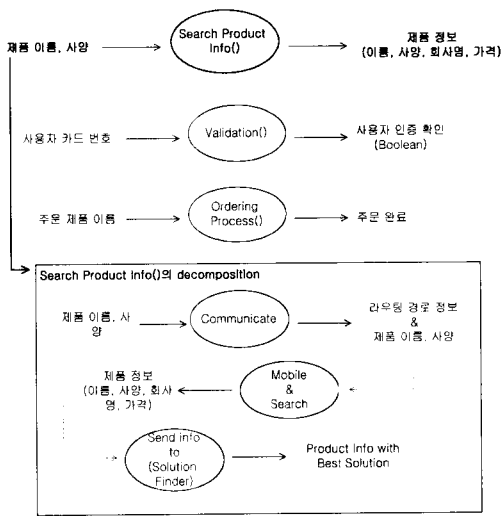

그림 12 EC예에서 제품 정보 탐색 에이전트의 기능 다이어그램

제품 정보 탐색 에이전트의 기능을 이로 표현하면 [그림 12]와 같다. 제품 정보 탐색 에이전트는 상품을 찾는 기능, 사용자 인증기능, 주문처리기능을 가지고 있으며 Search Product Info() 기능에 대해서는 좀 더 자세히 나타내었다. 즉, 상품을 찾기 위해 필요한 라우팅 경로를 알아내는 기능, DB를 이동하면서 필요한 정보를 찾는 기능, 찾은 정보를 솔루션 파인더 에이전트에게 보내어 최적의 결과를 얻어내는 기능을 DFD로 나타내었다.

4.5 에이전트 외부 모델

에이전트의 외부모델링은 에이전트의 이동과 관련된 이동성과 에이전트간의 메시지 교환을 모델링 하는 것이다. 이들에 대한 모델링을 전자상거래 예에 적용하면 다음과 같다.

4.5.1 에이전트 이동 모델링

에이전트의 이동성은 이동 에이전트가 작업을 수행하기 위해 이동하는 모습과 목적지를 결정하는 메커니즘을 보여주는데 이를 다이어그램으로 나타내면 [그림 13]과 같다.

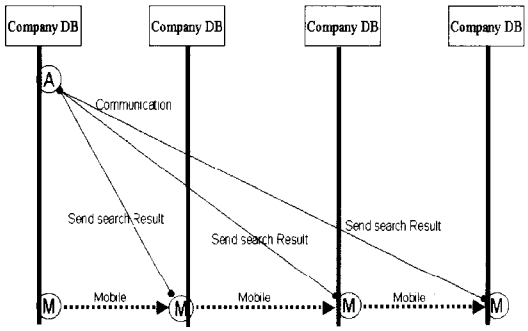

그림 13 EC예에서의 에이전트 이동성

여기서 *제품 정보 탐색 에이전트*가 이동 에이전트가 되고, *중재 에이전트*와의 통신을 통하여 이동경로를 알아내고 데이타베이스에서 찾은 결과를 보내준다.

4.5.2 에이전트 통신 모델링

에이전트의 통신 모델에서는 문제 영역에 존재하는 에이전트들 중에서 어떤 에이전트들 사이에 커뮤니케이션이 이루어지며, 그들이 주고받는 메시지의 형태는 어떤 것인지에 대한 정보를 보여주는데 이를 다이어그램으로 나타내면 [그림 14]와 같다.

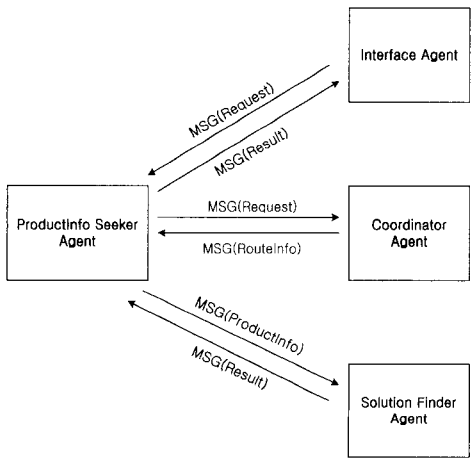

그림 14 EC예에서의 에이전트 통신

에이전트가 작업을 수행하면서 발생하는 결과 값과 자신에게 주어진 목표가 입력으로 들어가면, 내부의 처리 루틴에 의해 에이전트들이 통신하는 데 필요한 메시지가 생성된다. 여기서 에이전트가 사용하는 메시지 형식도 3개의 인자를 갖는데 이는 *sender_address, receiver_address, message*로 구성된다.

5. 결 론

본 논문에서는 실세계에서의 에이전트 추출과 이의 내부 및 외부 모델링을 통하여 에이전트 지향의 모델링 방법을 제안하였다. 에이전트와 객체들이 공존하는 세계에서 어떻게 에이전트화 되어야 할 부분을 선택해야 하는지에 대한 기준을 제시하였으며, *에이전트-클래스 다이어그램*을 통해 에이전트와 클래스가 공존하는 시스템의 정적인 면과 에이전트간의 상호 협력관계를 나타낼 수 있었다. 또한 목표 계층 다이어그램을 통해 각 목표에 해당하는 에이전트간의 협력관계를 분석할 수 있었으며, 자기정보를 통해 에이전트가 가져야 할 정보 중 온톨로지로 구축해야할 것과 아닌 것들을 구분하여 제시할 수 있었다. 계획 *시퀀스 다이어그램*을 통해 에이전트 내부에서 시간의 흐름에 따른 에이전트의 행동의 변화를 묘사할 수 있었으며, 기능을 통해 에이전트의 기능을 자세히 나타낼 수 있었다. 부가적으로 에이전트의 이동 속성과 여러 에이전트간의 통신 측면도 나타낼 수 있었다.

여기서 제안한 모델들은 각각에 대한 모듈을 만들고 이를 아키텍쳐에 매핑시킴으로써 구현될 수 있다. 자기정보 모델은 지식베이스 구축을 통하여 시스템 전체에 적용되는 정보들과 에이전트 각각의 필요한 정보들을 가질 수 있게 하고, 계획 모델은 에이전트의 목표를 성취할 수 있도록 순차적으로 일을 진행하고 관리하는 컨트롤 모델로 구현될 수 있다. 기능 모델은 계획 모델로부터 필요한 기능들을 뽑아내어 이를 실행하는 메소드 모듈 형태를 가질 것이고, 통신 모델과 이동모델 또한 메시지교환과 이동되는 장소에 대한 타입을 제시하고 관리하는 모듈로 구현될 것이다. 이러한 모델들을 구현한 뒤에는 목표모델을 검증과 테스트 모듈로 만들어 각 모델들이 목표 지향적으로 구현되었는지 검증해 볼 수 있을 것이다.

그러나 제시한 모델링 기법으로 앞에서 나열한 에이전트의 자율적인 특성을 포함한 모든 성질들을 효과적

으로 모델링 했는가에 대한 지속적인 고찰과 실제 에이전트 구현을 통한 검증 과정이 절실히 필요하다. 에이전트 추출 기준에 대한 타당성을 평가할 필요가 있다. 이를 위해 목표 지향의 모델링 방법에 대한 연구가 진행중이다. 이는 에이전트가 목표를 성취하려고 하는 능동적인 객체라는 개념에 근거한 것이다. [그림 15]는 에이전트 지향의 소프트웨어 개발 방법론에 대한 전체적인 면을 보여준다.

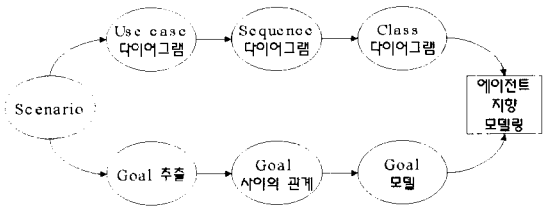

그림 15 에이전트 지향의 소프트웨어 방법론 개관도

앞으로는 우리가 제안한 모델링 방법에 기반하여 여러 가지 에이전트 지향의 문제영역에서의 시스템 구현을 통해 모델링 방법을 보완하고, 더 나아가 체계적인 에이전트 지향의 소프트웨어 개발 방법론에 관한 연구들을 단계적으로 지속해 나가고자 한다.

참 고 문 헌

[1] N. R. Jennings, K. P. Sycara, and M. Wooldridge A Roadmap of Agent Research and Development In Journal of Autonomous Agents and Multi-Agent Systems. 1(1), pages 7-36. July 1998.

[2] UMBC Lab for Advanced Information Technology, Intelligent Software Agents ,http://www.cs.umbc.edu/agents/

[3] M.Schroeder, Are Distributed objects agents?, *International Bi-Conference Workshop on AGENT-ORIENTED INFORMATION SYSTEMS (AOIS'99)*, 1999

[4] Paul Harmon and Mark Watson, Understanding UML : The Developers Guide, Morgan Kaufmann Publishers, 1998.

[5] Bran Selic, and Jim Rumbaugh, Using UML for Modeling Complex Real-Time Systems, 1998.

[6] M. Wooldridge. Agent-based Software Engineering. *In IEE Proceedings on Software Engineering*, 144(1), pages 26--37, February 1997.

[7] S.Russel and P.Norvig. Artificial Intelligence : A Modern Approach. Prentice-Hall, 1995.

[8] RAO.A.S and GEORGEFF, M.P.: 'Modeling rational agents within a BDI-architecture'. Proceedings of *Knowledge representation and reasoning* (KR&R-91), (Morgan Kaufmann Publishers, 1991), pp. 473-484

[9] David Kinny, Michael Georgeff, and Anand Rao. A methodology and modelling technique for systems of BDI agents. In W. van der Velde and J. Perram, editors, *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World MAAMAW'96*, (LNAI Volume 1038). Springer-Verlag: Heidelberg, Germany, 1996.

[10] Birgit Burmeister. Models and methodology for agent-oriented analysis and design. In K Fischer, editor, *Working Notes of the KI'96 Workshop on Agent- Oriented Programming and Distributed Sytstems*, 1996. DFKI Document D-96-06.

[11] Benjamin Falchuk and Ahmed Karmouch, "Visual Modeling for Agent-Based Applications," *IEEE computer*, December 1998.

[12] Shoham,Y., "Agent-Oriented programming," *Artificial Intelligence*, 1993, 60(1), pp. 51-92

[13] Thomas,S.R., PLACA, An Agent Oriented Programming Language, PhD Thesis, Stanford University, 1993

[14] A. Dardenne, A. van Lamsweerde and S. Fickas, Goal-directed Requirements Acquisition, Science of Computer Programming, 20, pp.3-50, 1993.

[15] T.R.Gruber, A Translation Approach to Portable Ontologies, Knowledge Acquisition, 5(2), 199-220, 1993.

[16] UMBC Lab for Advanced Information Technology, KIF(Knowledge Interchange Format), http://www.cs.umbc.edu/kse/kif/

[17] M.J.Kim, J.T.Kim, I.J.Park, S.J.Lee, and S.Y.Park, Agent-based Software Analysis Method in Distributed Environment, *Fuzzy-IEEE'99*, September 1999.

김 민 정

1998년 2월 서강대학교 공과대학 전자계산학과 학사. 2000년 서강대학교 공과대학 컴퓨터학과 소프트웨어 공학 석사. 현재 ETRI 컴퓨터, 소프트웨어기술연구소 S/W공학연구부 컴포넌트공학연구팀 연구원. 관심분야는 에이전트 지향 소프트웨어 공학(AOSE), 컴포넌트 기반 개발(CBD).

이 승 연

1999년 2월 서강대학교 공과대학 컴퓨터학과 학사. 현재 서강대학교 공과대학 컴퓨터학과 소프트웨어 공학 석사과정. 관심분야는 소프트웨어 아키텍쳐, 소프트웨어 개발 방법론, 에이전트 지향 소프트웨어 공학(AOSE).

박 근 하

1999년 2월 서강대학교 공과대학 컴퓨터학과 학사. 현재 서강대학교 공과대학 컴퓨터학과 소프트웨어 공학 석사과정. 관심분야는 시스템 분석 및 설계, 소프트웨어 개발 방법론, 에이전트.

박 원 영

2000년 2월 서강대학교 공과대학 컴퓨터학과 학사. 현재 서강대학교 공과대학 컴퓨터학과 소프트웨어 공학 석사과정. 관심분야는 Pattern Language, 소프트웨어 아키텍쳐, 다중 에이전트 시스템.

박 수 용

정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 1 호 참조