온톨로지와 사용자 프로파일을 적용한 지능형 서비스 에이전트

(Intelligent Service Agents using User Profile and Ontology)

김 제 민 † 박 영 택 ††
(Je-Min Kim) (Young-Tack Park)

요 약 최근, "유비쿼터스 컴퓨팅"이라는 지능형 서비스 프레임워크가 제안되면서 적응형 에이전트 시스템의 필요성이 점점 증가되기 시작됐다. 본 논문에서는 유비쿼터스 컴퓨팅 시스템이 사용자에게 적절한 서비스를 제공하도록 도와주는 지능형 서비스 에이전트를 제안한다. 사용자에게 적절한 유비쿼터스 서비스를 제공하기 위해서는, 각각의 유비쿼터스 서비스 시스템 내에서의 상황 정보(Context Information) 차이를 조절하고 사용자의 취향을 서비스에 반영해야 한다. 따라서 다음 3가지 부분에 중점을 두어 연구를 진행하였다. 첫째, 적절한 다중 에이전트 프레임워크 - 에이전트간의 커뮤니케이션 이해와 추론엔진의 적용, 둘 째, 유비쿼터스 컴퓨팅 환경 내에 존재하는 다양한 상황 정보(Context information)를 효과적으로 표현하는 유비쿼터스 온톨로지 - 에이전트간의 상황 정보 공유와 이해, 마지막으로 유비쿼터스 시스템에 적용되는 사용자 프로파일 구축 방법에 대해 연구 하였다. 본 논문에서 제안하는 지능형 서비스 에이전트는 사용자 취향에 따라 적절한 서비스를 제공하는 적응형 유비쿼터스 서비스 시스템 구축을 가능하게 한다.

키워드 : 지능형 서비스 에이전트, 에이전트 커뮤니케이션 언어, 온톨로지, 사용자 프로파일, 추론 엔진

Abstract Recently, new intelligent service frameworks, such as ubiquitous computing are proposed. So, the necessity of adaptive agent system has been increased. In this paper, we propose an intelligent service agent to help that ubiquitous computing system offer user suitable service in ubiquitous computing environment. In order to offer user suitable uT-service, an intelligent service agent mediates the gap between the context information in uT-service system, and user preference is reflected in it. Therefore, we focus on following three components; the first is suitable multi agent framework - agent communication analysis and applicable method of inference engine, the second is uT-ontologies to describe various context information - context information sharing between agents and context information understanding between agents, the third is learning method of user profile to apply in uT-service system. This approach enables us to build adaptive uT-service system to offer suitable service according to user preference.

Key words : Intelligent Service Agent, ACL - Agent Communication Language, Ontology, User Profile Inference Engine

1. 서 론

최근, "유비쿼터스 컴퓨팅"이라는 지능형 서비스 프레임워크가 제안 되고 있다. 유비쿼터스 컴퓨팅(Ubiquitous Computing)은 수많은 지능형 컴퓨터들이 유기적으로 연결되어 언제 어디서나 사물(Object)을 인식하고 사람들에게 필요한 정보나 서비스를 제공을 목적으로 하며[1], 유비쿼터스 컴퓨팅 환경은 다양한 형태의 지능형 컴퓨터가 현실 세계와 효과적으로 결합되어 언제 어디서나 컴퓨팅 자원을 활용할 수 있는 환경을 제공한다. 어플리케이션 간의 기능 전달 및 자율적인 활동이 가능하도록 구현된 소프트웨어 프로세스인 에이전트는 유비쿼터스 컴퓨팅에 있어 매우 중요한 역할을 담당한다. 이들은 에이전트 언어(Agent Communication Language)를 사용하여 서로 상호 작용을 한다[2].

유비쿼터스 컴퓨팅 환경 내에서 에이전트들이 효과적인 기능을 발휘하기 위해서는 각 에이전트간의 원활한

· 본 논문은 숭실대학교의 지원을 받았습니다.
† 학생회원 : 숭실대학교 컴퓨터학과
kimjemins@hotmail.com
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
park@computing.ssu.ac.kr
논문접수 : 2005년 3월 4일
심사완료 : 2006년 10월 24일

상호작용과 사용자 개개인에 적합한 서비스를 제공하기 위한 추론-행위 모듈이 필요하다. 본 논문에서는 이러한 두 가지 사항을 고려하여 구현된 JADE기반 지능형 서비스 에이전트를 제안한다.

사용자에게 적절한 유비쿼터스 서비스를 제공하기 위해서, 지능형 서비스 에이전트는 각각의 유비쿼터스 서비스 시스템 내에서의 상황 정보(Context Information) 간의 차이를 조절하고 사용자의 취향을 서비스에 반영한다. 이러한 기능을 가지는 지능형 서비스 에이전트를 제안하기 위해서 다음 3가지 부분에 중점을 두었다.

첫째, 적절한 다중 에이전트 프레임워크를 선택이다. 적절한 선택을 위해 에이전트의 활동 및 관리뿐만 아니라, 서비스 추론 엔진과의 상호작용 고려한다. 지능형 서비스 에이전트가 사용자에게 적절한 서비스를 제공해 주기 위해서는 사용자 프로파일을 기반으로 동작하는 추론 엔진이 적용되어야 한다. 추론 엔진 내에는 사용자 프로파일로부터 전달되어 각 사용자에 적용되는 서비스 규칙(rule)을 포함한다. 둘째, 유비쿼터스 컴퓨팅 환경 내에 존재하는 다양한 상황 정보(Context Information)를 효과적으로 표현하는 유비쿼터스 온톨로지의 정의다. 유비쿼터스 컴퓨팅 환경 내에서 상황 정보(Context Information)에 대한 중요성이 점점 증가하고 있는데, 상황(context)은 사용자와 유비쿼터스 서비스 시스템간의 상호작용을 위해 필요한 사용자 주변의 환경, 객체 또는 사용자의 위치에 관한 정보를 의미한다[3]. 상황 정보는 유비쿼터스 컴퓨팅 환경 내에 존재하는 다양한 센서에 의해 다양한 형태로 획득 된다. 서로 다른 포맷으로 상황 정보를 인지하는 에이전트들 간의 일관성 있는 상황 정보의 공유를 위해, 모든 상황 정보들은 일관성 있는 개념(Concept)으로 정의되어야 한다. 유비쿼터스 온톨로지는 모든 에이전트들이 상황 정보들을 공유할 수 있도록, 상황 정보에 대한 개념을 형식적이고, 명백하게 명시한 스키마를 제공하기 때문에, 에이전트간의 상황 정보 공유 및 관리를 수월하게 하게 해준다. 마지막으로, 유비쿼터스 시스템에 적용되는 사용자 프로파일 구축 방법이다. 사용자의 취향에 따라 적합한 유비쿼터스 서비스를 제공하기 위해서는 정확한 사용자 프로파일을 구축하는 것이 중요하다. 기존의 유비쿼터스 컴퓨팅 서비스를 지원하는 다중 에이전트 시스템은 사용자 프로파일을 각 사용자의 개인 정보가 담긴 홈페이지에서 제공 받고 있다. 이러한 방식은 사용자가 매번 자신의 서비스 취향을 직접 홈페이지에 기록하기 때문에 그다지 효율적이지 못하다고 할 수 있다. 따라서 지능형 서비스 에이전트에 귀납적 기계학습을 적용하여 시스템 스스로 사용자 프로파일을 구축하는 방안을 연구한다.

본 논문의 본문에서는 제안된 지능형 서비스 에이전트의 유비쿼터스 컴퓨팅 환경에서 에이전트들 간의 정확한 의미 전달을 통한 상호작용과 사용자 프로파일을 생성 및 서비스 제공 에 대해 기술한다. 2장에서는 관련 연구를 설명하고, 3장에서는 JADE 기반 다중 에이전트 플랫폼을, 4장에서는 유비쿼터스 온톨로지를, 5장에서는 사용자 프로파일 구축방안에 대해 기술한다. 6장에서는 제안된 지능형 서비스 에이전트의 핵심인 에이전트간의 상호작용과 사용자 프로파일 기반의 서비스 추론에 대한 간단한 실험 결과를 설명하고, 마지막 장에서는 연구 결과 및 향후 연구되어야 할 부분에 대해 기술한다.

2. 관련 연구

마크 와이저에 의해 "유비쿼터스 컴퓨팅"이라는 새로운 개념이 정립된 이래로 다중 에이전트 기술과 온톨로지를 유비쿼터스 컴퓨팅 서비스에 적용하려는 연구들이 활발히 진행되고 있다. 온톨로지 기반의 다중 에이전트 시스템을 유비쿼터스 서비스 시스템에 적용한 대표적인 연구로는 UMBC(University of Maryland Baltimore County)의 CoBrA[4,5]와 UIUC(University of Illinois at Urbana-Champaign)의 GAIA[6]가 있다.

CoBrA는 지능형 회의실, 집, 사무실과 같은 지능형 공간을 지원하는 에이전트 기반 시스템의 전체적인 구조를 총칭하는 말이다. 이 구조의 가장 중심이 되는 것은 컨텍스트 브로커(Context Broker)라고 불리는 지능형 에이전트다. 컨텍스트 브로커는 유비쿼터스 컴퓨팅 환경 내에서 에이전트와 장치들이 서로 상황 정보들을 공유할 수 있도록 상황 정보들을 관리하고, 유비쿼터스 서비스를 제공 받는 사용자들의 개인 정보를 보호한다. CoBrA의 가장 큰 특징은 유비쿼터스 컴퓨팅 환경에서 발생하는 모든 상황(Context)들을 OWL(Web Ontology Language)[7] 기반의 온톨로지로 표현한다는 점이다. 이는 에이전트 간의 상황 정보 공유를 가능하게 하고, 상황 정보를 이용한 서비스 추론이 가능하게 한다.

GAIA는 유비쿼터스 컴퓨팅 시스템을 위한 미들웨어다. GAIA는 물리적 공간 및 물리적 공간 안에 존재하는 모든 장치들을 가상공간 안에 포함함으로써 물리적 공간에서 발생하는 모든 서비스들을 관리한다. 즉, 서비스 관리와 유비쿼터스 컴퓨팅 환경 내에 존재하는 객체들(사람, 사물)의 표현 및 검색에 대한 기본적인 기능을 수행하기 때문에 GAIA를 기반으로 작동하는 모든 유비쿼터스 서비스 시스템들 간의 인터페이스를 제공해 줄 수 있다. GAIA의 가장 큰 특징은 역시 유비쿼터스 컴퓨팅 환경 내에 존재하는 모든 객체와 서비스들을 온톨로지로 표현함으로써 유비쿼터스 서비스 시스템들 간의 객체 및 서비스 정보에 대한 공유가 가능하다. GAIA의 온톨로지는 DAML+OIL로 작성 되었으며, 온톨로지 서

버를 이용하여 온톨로지로 표현된 객체 및 서비스 정보를 관리한다.

CoBrA와 GAIA처럼 많은 시스템들이 온톨로지를 사용하여 유비쿼터스 컴퓨팅 환경 내에 모든 상황 정보들을 서로 이해하고 공유할 수 있도록 연구 되고 있다. 이는 본 논문의 첫 번째 목적과 일치하는 부분이다. 그러나 대부분의 시스템들은 온톨로지로 표현된 상황 정보를 사용하여 보편적이고 간단한 서비스(사람이 화장실에 있으면 조명 장치를 작동시킨다.)를 제공하도록 구현되었다. 따라서 본 논문에서는 이 전에 연구되었던 시스템들처럼 온톨로지를 기반으로 에이전트간의 상호작용(상황 정보 공유 및 메시지 전달)을 원활히 하고, 이에 덧붙여, 사용자 개개인에 적합한 서비스를 제공하기 위한 추론엔진의 적용 및 사용자 프로파일을 자동으로 구축하는 지능형 서비스 에이전트를 제안하고자 한다.

3. JADE 기반 다중 에이전트 플랫폼

본 논문의 목적은 에이전트간의 원활한 상호작용과 에이전트가 사용자에게 적합한 서비스를 제공하기 위한 지능형 서비스 에이전트를 연구하는 것이다. 에이전트 플랫폼은 크게 에이전트와 유비쿼터스 온톨로지, 추론엔진으로 구성된다. 에이전트 플랫폼이 제공하는 가장 기본적인 기능은 각 에이전트 관리, 에이전트간의 커뮤니케이션 제공, 에이전트의 기능을 등록하는 에이전트 디렉토리 서비스를 제공함으로써 에이전트들이 활동할 수 있는 장소를 제공한다. 본 논문에서 제안하는 JADE 기반 에이전트 플랫폼의 기본적인 기능은 FIPA[8]의 표준을 따르고 있다. 본 장에서는 FIFA의 표준을 기반으로 서비스 추론 엔진이 적용된 다중 에이전트 플랫폼을 구성하기 위해 필요한 컴포넌트에 대해 설명한다.

3.1 에이전트 관리 컴포넌트

에이전트 관리 컴포넌트(Agent Management Component)는 에이전트의 사용과 접근을 총괄하는 상위 에이전트다. 즉 에이전트가 생성되면 에이전트 관리 컴포넌트는 에이전트 식별자(AID)를 할당하며, 이 식별자를 통해서 에이전트들의 실행 주기(life-cycle)를 제어하게 된다. 각 에이전트는 에이전트의 실행 주기에 따라 다음 표 1 같은 상태(state)중의 하나를 가지게 된다.

3.2 에이전트 등록 컴포넌트

에이전트 등록 컴포너트는 일종의 에이전트들을 위한 서비스 옐로우 페이지 역할을 한다. 옐로우 페이지는 인터넷상에서 생활서비스 혹은 제품을 제공하는 회사나 전문직종의 전화번호를 안내하는 전화번호부 이거나 사이트 목록을 분야별로 정리해 놓은 자료를 지칭하는 용어다. 즉, 에이전트가 어떠한 서비스를 요구할 때, 필요한 서비스를 제공하는 에이전트를 에이전트 디렉토리와 서비스 디렉토리에서 검색한 후, 검색된 에이전트에게 서비스를 요구하는 메시지를 보낸다.

표 1 에이전트의 실행 주기

에이전트 상태	설 명
초기화	에이전트가 새로 생성된 상태. 그러나 에이전트 관리 시스템에게 AID를 할당 받지 않은 상태이기 때문에 다른 에이전트와 상호작용이 불가능하다.
활성화	생성된 에이전트에 에이전트 관리 시스템이 AID를 할당한다. 에이전트는 고유의 에이전트 이름과 주소를 갖게 되며 시스템 안의 모든 에이전트와 상호작용이 가능하다.
중지	에이전트 관리 시스템의 스케줄링에 의해 에이전트의 동작이 잠시 멈춘 상태이며, 중지 상태인 에이전트는 자신의 행위 모듈을 실행할 수 없다.
대기	에이전트 관리 시스템의 스케줄링에 의해 에이전트가 블록킹된다. 블록킹 상태의 에이전트는 특정 태스크가 실행될 때까지 대기한다.
삭제	등록된 에이전트가 삭제된 상태. 이때 에이전트 관리 시스템은 삭제될 에이전트로부터 AID를 반환 받는다.

에이전트 등록 컴포넌트는 크게 에이전트 디렉토리와 서비스 디렉토리로 나뉜다. 에이전트 디렉토리는 에이전트들이 자신에 대한 간단한 정보를 저장할 수 있는 위치를 제공한다. 이러한 정보는 에이전트-이름과 에이전트-위치로 구성되어, 에이전트는 자신이 존재하는 장소(플랫폼 또는 컨테이너)를 등록할 수 있다. 서비스 디렉토리는 에이전트가 자신의 서비스에 대한 정보를 등록할 수 있는 기능을 제공한다. 이러한 정보는 서비스 이름, 서비스 타입, 서비스 위치(서비스와 연결된 에이전트 위치)로 구성되어, 에이전트와 에이전트가 제공하는 서비스를 연결해 줌으로써 필요한 서비스를 제공하는 에이전트를 검색할 수 있다.

3.3 에이전트 메시지 전달

유비쿼터스 컴퓨팅 환경에서는 하나의 에이전트로는 모든 서비스를 수행할 수 없다. 처리하지 못하는 서비스의 수행을 위해 다른 에이전트의 도움을 필요로 할 때 에이전트는 메시지 교환을 통해 필요한 서비스를 제공하는 다른 에이전트에게 서비스를 요구할 수 있다. 즉, 에이전트는 자신이 필요한 서비스를 제공하는 에이전트들을 검색한 후, 서비스가 필요한 에이전트는 서비스가 실행 가능한지, 실행되고 있는지, 실행해줄 수 있는지에 대해 메시지를 보내게 되고, 서비스를 제공하는 에이전트는 적합한 응답 메시지를 보내준다. 유비쿼터스 컴퓨팅 환경에서 각 서비스 시스템은 에이전트간의 메시지 전달을 통해 필요한 서비스를 제공하는 에이전트를 검색한 후, 사용자에게 서비스를 제공한다.

에이전트 통신 언어(ACL-Agent Communication

표 2 ACL 메시지에서 사용하는 수행문

Communicative Act	설명	예.
Query	다른 에이전트가 이행하고 있는 행위가 사실 또는 거짓인지에 대한 질문을 표현	텔레비전이 켜져 있나?
Request	다른 에이전트에게 특정 행위를 이행하도록 요구를 표현	텔레비전을 켜라!
Agree	앞으로 어떠한 행위를 이행하는데 있어서 동의를 표현	오케이! 텔레비전을 켜겠다.
Inform	에이전트가 다른 에이전트에게 현재 상태 또는 조건이 사실임을 표현	텔레비전이 켜졌다.
Failure	에이전트가 시도는 하였으나 실패한 행위에 대한 표현	나는 텔레비전을 켤 수 없다.
Refuse	에이전트가 이행할 수 있는 행위에 대한 거부를 표현	나는 텔레비전을 켜기 싫어!
Subscribe	에이전트가 이행하고 있는 행위의 상태가 바뀔 때마다 통보하는 것에 대한 표현	텔레비전이 켜졌을 때 말해줘!
CFP	어떠한 행위를 이행하기 위해, 그 행위를 필요로 하는 에이전트 호출에 대한 표현	누가 텔레비전을 보고 싶나요?
Propose	일정 조건에 맞는 행위가 실행 가능함을 표현	내가 텔레비전을 켜줄 수 있어.
Not-understood	에이전트가 이행할 수 없는 행위를 요구받았을 경우, 요청 받은 행위를 이행불가를 표현	텔레비전이 머지? 무슨 말인지 이해를 못하겠어.

Language)를 이용한 에이전트 간의 통신은 메시지 전달(Message passing)이나 공유 메모리(Shared memory) 방법을 이용할 수도 있고 다른 에이전트의 메소드(Method)를 불러 수행하기도 한다. 본 논문에서는 제안한 시스템은 FIFA에서 정의한 ACL 메시지를 사용하여 에이전트간의 통신을 통해 사용자에게 서비스를 실행하도록 설계되었다. ACL 메시지는 메시지의 특성을 정의하는 파라메터를 포함하고 있으며, 이 중 중요한 파라미터로는 수행문(Performative), 보내는 에이전트(Sender), 받는 에이전트(Receiver), 메시지 내용(Contents)이다. 특히 수행문은 서비스 요구나 거절과 같이 메시지의 목적을 결정하며, 표 2는 ACL 메시지의 수행문들과 예를 기술한 것이다.

아래에 서술한 ACL메시지는 userAgent라는 에이전트가 tvAgent라는 에이전트에게 텔레비전을 켜달라고 요청하는 것을 나타낸 예다. 여기서: CONTENT (turn on tv-ooa)는 서비스 요청 내용이 된다. "@"의 앞부분은 에이전트 이름을 나타낸 것이며, 뒷부분은 에이전트가 위치한 에이전트 플랫폼의 이름과 포트 번호 및 에이전트 플랫폼이 존재하는 서버 이름을 나타낸 것이다. 아래의 메시지는 메시지를 구성하는 파라미터 중의 가장 중요한 수행문(Performative), 보내는 에이전트(Sender), 받는 에이전트(Receiver), 메시지 내용(Contents)만을 사용하여 서비스 요청메시지를 구성하였는데, 에이전트가 메시지 전달을 통해 사용자에게 서비스를 제공하기 위해서는 기본적으로 이 4개의 파라미터를 기본적으로 가지고 있어야 한다.

ACL 메시지의 예

```
(Request
: Sender userAgent@basePlateform:1099
        /friend
: Receiver tvAgent@basePlateform:1099
  /friend
: Content
      (turn on tv-ooa)
)
```

3.4 서비스 추론 에이전트

에이전트는 크게 에이전트 생성 모듈과 에이전트 행위 모듈 두 가지로 나누어진다. 에이전트 생성 모듈은 에이전트 플랫폼에 새로운 에이전트를 생성하고 소멸하며, 에이전트 식별자(AID)를 할당 및 반환하는 역할만을 담당한다. 때문에 각각의 에이전트들은 자신만의 기능을 수행할 수 있는 모듈이 필요하며, 이러한 역할을 하는 것이 에이전트 행위 모듈이다. 본 논문에서 제안하는 지능형 서비스 에이전트의 목적중의 하나는 사용자 프로파일 기반으로 사용자에게 적절한 서비스를 제공하는 것이다. 이를 위해서는 사용자의 취향이 반영된 사용자 프로파일을 기반으로 동작하는 추론 엔진이 에이전트 행위 모듈에 적용되어야 한다. 추론 엔진이 적용된 행위 모듈은 술어(Predicate)로 작성된 규칙(Rule)과 사실(Fact)을 바탕으로 동작한다. 이러한 사실은 각 사용자 프로파일에 기록되기 때문에 각각의 사용자 취향에 맞는 서비스를 추론할 수 있다.

그림 1은 추론 엔진이 적용된 에이전트 행위 모듈의 구조를 나타낸 것이다. 유비쿼터스 컴퓨팅 환경 내에서의 상황 정보는 상황 정보 데이타베이스 저장되며, 상황 정보 사실 추출기에 의해 추론 엔진이 추론에 사용하는 사실(Fact)로 변환 된다. 추론 엔진을 구현할 때 가장 중요한 것은 지식 혹은 사실을 표현하는 방법이다. 사람이 사실을 표현할 때 가장 쉬운 방법이 일상적으로 사

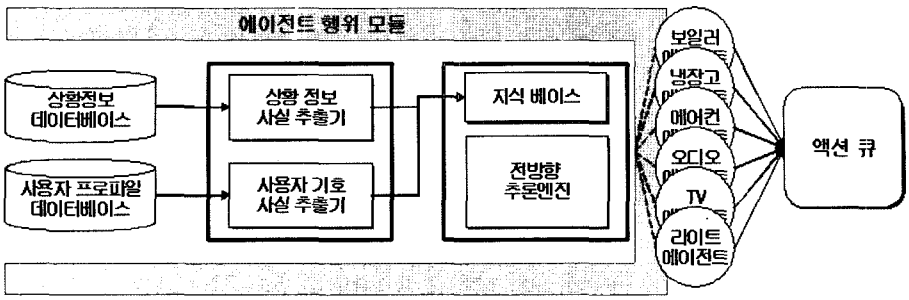

그림 1 추론엔진이 적용된 에이전트 행위 모듈 구조

용하는 말이다. 트리플에 의해 표현되는 상황 정보는 주어가 개별 개체인 하나의 문장 구조를 띠고 있기 때문에 술어(Predicate)로 표현이 가능하다.

추론 엔진이 사용자에 따라 적절한 서비스를 추론하려면 상황 정보뿐만 아니라 사용자 프로파일에 표현된 사용자 기호 정보에 대한 사실을 지식 베이스에 저장하고 있어야 한다. 사용자 기호 추출기는 사용자 프로파일 데이타베이스에 저장되어 있는 사용자 프로파일의 내용을 다음과 같은 술어 형태로 변환한다.

지식 베이스에 사실로 저장된 사용자 기호의 예

```
(userPreference (:NAME "Kim00a-1")
(startPtime "1700")
(endTtime "1800")
(lux 70)
(location livingRoom00a-1)
(Activity enter))
(userPreference (:NAME "Kim00a-1")
(startPtime "1700")
(endTtime "1800")
(lux 30)
(location livingRoom00a-1)
(Activity watchMovie)
```

위의 예제는 "Kim00a-1"이라는 사용자 식별자를 가진 사용자의 장소와 무드에 따른 형광등 조도에 관한 기호를 나타낸 것이며, "Kim00a-1이 거실에 17부터 18시 사이에 있고 형광등 조도가 70룩스다."와 "Kim00a-1이 거실에 17부터 18시 사이에 있고 집안의 무드가 영화를 보는 분위기이며 형광등 조도가 30룩스다."라는 사실을 의미한다.

본 논문을 위해 구축된 추론 엔진은 작업 메모리에 적재된 상황 정보와 사용자의 기호를 바탕으로 규칙 베이스 내에 정의된 서비스 규칙을 이용하여 사용자에게 적합한 서비스를 추론하며, 추론된 결과를 ACL 메시지로 만든 후 서비스를 요청한 에이전트에게 전송한다. 본 추론 엔진이 선택한 추론 전략은 기본적으로 전방향 추론(Forward chaining)[12]이다. 전방향 추론은 A → B 즉 조건 A가 지식베이스에 있으면 B를 수행한다. 서비스 규칙은 크게 상황 정보와 사용자의 기호를 바탕으로 추론된 서비스를 지식베이스에 저장하는 서비스 추론 규칙, 추론된 서비스를 ACL 메시지로 변환하는 규칙, 서비스를 요청한 에이전트에게 서비스 내용이 담긴 ACL메시지를 전송하는 규칙으로 구성된다.

서비스 규칙의 예

```
(defrule tv-on01 ""
 (object (hasID ?name)(hasActivity ?act))
 (object (:NAME ?ts)(endTime ?etime)
  (startTime ?stime))
 (object (:NAME ?act)(actedByPerson
  ?name)(actedAtTime ?ts)
(actedInLocation ?loc))
 (userPreference (:NAME ?name)
  (startPtime ?pstime&:(<= ?pstime
  ?stime))
   (endTime ?petime&:(>= ?petime
    ?etime)) (lux ?lux)(location ?loc))
    =>
  (assert (light (status on)(lux ?lux)))
(defrule send-inform ""
 (MyAgent(name ?n)) (receiver ?receiver)
 ?m <-(light (status ?status)(lux ?lux))
 =>
 (assert(ACLMessage (communicative-act
  INFORM)
  (sender ?n) (receiver ?receiver)
  (content (str-cat "(light(status "
   ?status ") (lux " ?lux "))")))))
  (retract ?m))
```

```
(defrule send-a-message ""
(MyAgent (name ?n)), ?m<-(ACLMessage
 (sender ?n))
 =>
(send ?m))
```

위의 서비스 규칙 예제는 거실 형광등을 담당하는 형광등 에이전트의 행위 모듈에 적용되는 서비스 규칙의 일부분을 나타내고 있으며, 기술되어 있는 서비스 추론 규칙은 "?name인 사람이 ?loc인 곳에서 ?pstime과 ?petime 사이에 ?act라는 행위를 하면 형광등 조명은 ?lux 였기 때문에, 사용자 식별자가 ?name인 사람이 위치가 ?loc인 곳에서 ?stime과 ?etime 사이에 ?act라는 행위를 하면, 형광등의 작동상태를 on으로 하고,' 조도를 ?lux로 조정하라는 사실을 지식 베이스에 저장"이라는 의미를 가진다. 이처럼 추론된 서비스는 지식 베이스의 사실로 저장되어 있다가 메시지 변환 규칙과 메시지 전송 규칙에 따라 서비스를 요청한 에이전트에게 서비스 내용을 전송한다. 이렇게 구축된 추론 엔진을 에이전트 행위 모듈에 적용함으로써 각각의 에이전트들은 사용자 프로파일을 기반으로 적절한 서비스를 추론하고, 추론된 서비스메시지를 다른 에이전트(서비스를 요청한 에이전트 또는 서비스 액션 큐)에게 전송하기 때문에, 본 논문에서 제안한 지능형 서비스 에이전트는 사용자에게 적절한 서비스를 제공하는 것이 가능하다.

4. 상황 정보 관리를 위한 유비쿼터스 온톨로지

본 논문에서 제안하는 지능형 서비스 에이전트는 효율적으로 상황 정보를 공유할 수 있도록 시멘틱 웹 온톨로지를 사용한다. 온톨로지를 적용하면 지능형 공간에 존재하는 사람, 장치, 에이전트, 시간, 공간 등에 대한 모델을 제공하기 때문에, 상황 정보의 일관성 있는 관리와 공유가 가능하다.

온톨로지는 단어와 관계들로 구성된 사전으로서 어느 특정 도메인에 관련된 단어들을 계층적 구조로 표현하고, 이를 확장하여 표현할 수 있는 추론 규칙을 포함한다. 온톨로지를 지능형 서비스 에이전트에 적용하는 가장 큰 이유는 서로 다른 센서가 같은 객체에 대해서 서로 다른 단어나 식별자를 사용하여 센싱할 경우에 이를 해결해준다. 예를 들어 person과 people은 같은 의미를 가진다. 이 두 단어를 사용하려는 에이전트가 있다면, 이 두 단어가 같은 의미를 가진다는 사실을 인식해야 하며, 이는 온톨로지를 통해 이루어진다. 이렇듯이 온톨로지는 여러 시스템들 사이의 지식 공유를 가능하게 한다. 제안하고 있는 시스템에 적용하기 위해 구축된 온톨로지는 웹 온톨로지 언어인 OWL로 작성되었으며 지능형 홈(Intelligence Home)을 위해 여러 가지 상황 모델(행위, 에이전트, 장치, 시간, 공간 등등)을 제공한다. 따라서 이전에 구축되어 공유 가능한 온톨로지(Friend of s friend, DAML Time, Spatial ontology)들이 적용되어있으며, 물리적인 공간, 시간, 사람, 에이전트, 장치, 행위에 대한 전형적인 개념과 관계를 정의한다. 본 연구를 위해 구축한 온톨로지는 다음과 같이 구성되어 있다.

- 행위 온톨로지 - 구축된 유비쿼터스 온톨로지중 핵심 부분에 해당하며 사람의 행위에 대한 상황(Context)을 정의한다. 행위 온톨로지는 행위의 주체가 되는 사람, 행위에 필요한 장치, 행위가 일어난 장소, 행위가 일어난 시간 등을 속성(Property)으로 정의 하고 있으며, 행위의 주체가 되는 사람을 정의한 Person 온톨로지, 장소를 정의한 Location 온톨로지, 장치를 정의한 Device 온톨로지, 행위가 일어난 시간을 정의한 Time 온톨로지의 개체(Instance)들을 속성 값으로 가지고 있기 때문에 누가, 언제, 어디서, 어떤 장치를 사용하여 행위를 취했는지 일관성 있게 표현할 수 있다.
- Person 온톨로지 - Person 온톨로지는 사람의 정보를 상황 정보로 표현하기 위한 클래스 형태로써 이름, 성별, 나이, 식별자, 현재 사람이 위치한 장소의 프로퍼티를 가진다. 따라서 사람이 가지고 있는 여러 가지 속성에 대한 전형적인 어휘들로 구성된다. 이 온톨로지는 FOAF(Friend of s friend) 온톨로지를 기반으로 구축됐다.
- Device 온톨로지 - Device에 대한 정보와 실행 가능한 서비스, 위치정보, 사용자 등등에 대한 정보를 정의한다. 따라서 Device 온톨로지는 Object라는 메인 클래스를 지니며, 서비스형태, 상태, 위치한 장소, 제조사, 식별자의 프로퍼티를 갖는다. Object 클래스는 하위 클래스로 appliance와 furniture를 갖는다.
- Temporal 온톨로지 - 유비쿼터스 환경에서 모든 개체의 상태정보와 위치정보에 시간 개념을 적용시키기 위한 온톨로지다. Temporal 온톨로지는 Time이라는 클래스로 표현되는데, 이 클래스는 특정 시간과 시간 간격에 대한 정의 및 이들간의 관계를 표현하는 전형적인 어휘들로 구성된다. 따라서 Temporal 온톨로지의 시간 관계를 이용하여 서로 다른 시간에 발생하는 이벤트의 시간적인 속성을 정의할 수 있다. 이 온톨로지는 DAML Time 온톨로지를 기반으로 구축됐다.
- Spatial 온톨로지 - Spatial 온톨로지는 지능형 홈 내에서 객체의 위치를 일정한 장소에 매치시키위한 전형적인 어휘 및 기호들로 구성된다. 유비쿼터스 환경에서 장소에 대한 온톨로지는 매우 중요한 요소이다. 사용자의 행동으로 실시간 발생하는 Context 정보들이 Location에 대한 정의가 없다면 서비스 에이전트

로 하여금 적합한 서비스를 수행하는 것은 불가능 하다고 볼 수 있다. 따라서 지능형 홈을 위해 구축된 Spatial 온톨로지는 Space라는 최상위 클래스를 생성하여, 하위 클래스로 House 클래스를 가지며, House 클래스는 Entrance, Kitchen, Livingroom, Bathroom, Bedroom, Dressingroom, Boilerroom, Veranda등의 하위 클래스를 갖는다.

- Service 온톨로지 - 유비쿼터스 환경에 존재하는 각각의 장치의 동작들은 사용자에게 적합한 서비스를 제공하기 위해 정형화된 형식으로 표현돼야 한다. 따라서 Service 온톨로지는 클래스 형태로써 Following Service, Heating Service, LC Service 기능을 하는 하위 클래스를 갖는다.

유비쿼터스 컴퓨팅 환경에서 발생하는 유저의 상황은 곳곳에 장치된 센서에 의해 감지되어, 다중 에이전트 시스템에 연결된 컨텍스트 브로커에 의해 상황 정보가 표현된 온톨로지 개체(Ontology Instance) 파일로 실시간 생성되어 상황정보 데이타베이스에 저장된다.

그림 2는 "Kim00a-1이 2000년 1월 3일 17시 9분에 거실로 들어왔다"라는 상황 정보를 온톨로지 개체 파일로 표현된 것이다. 컨텍스트 브로커는 본 논문의 범위에서 벗어나므로 자세한 설명은 생략하겠다.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="" />
 </owl:Ontology>
  <Activity rdf:ID="enter">
   <actedAtTime>
    <Time rdf:ID="timesequence00t-1">
     <startTime rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >20000103-1709</startTime>
     <endTime rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >20000103-1710</endTime>
    </Time>
   </actedAtTime>
   <actedByPerson rdf:resource="#Kim00a-1" />
   <actedInLocation>
    <LivingRoom rdf:ID="livingRoom00a-1" />
   </actedInLocation>
  </Activity>
</rdf:RDF>
```

그림 2 상황정보를 표현한 온톨로지 개체 파일

본 논문에서 제안한 지능형 서비스 에이전트는 사용자의 기호에 맞는 서비스를 제공하기 위해 추론 엔진을 적용한 행위-모듈을 사용한다. 추론 엔진이 서비스를 추론하기 위해서는 사용자의 기호를 규칙(Rule)으로 표현해야 함은 물론이고 온톨로지 개체로 표현된 상황 정보를 지식 베이스(Knowledge Base)에 사실(Fact)로 저장해야 한다. 그러나 그림 2와 같이 온톨로지 개체들을 표현한 온톨로지 파일들은 추론 엔진이 사실(Fact)로 처리하는데 큰 어려움이 있다. 그래서 본 논문에서 제안한 다중 에이전트 시스템은 상황 정보가 표현된 온톨로지 개체 파일들을 [subject, property, object]로 구성된 트리플 구문[10]으로 변환하여 사용하도록 구현되었다.

트리플 구문은 상황 정보의 속성 및 관계를 명확하게 표현해주면서, 추론 엔진의 사실을 구성하기 원활하게 한다. 그림 3은 그림 2의 온톨로지 개체 파일을 트리플 구문으로 변환한 것을 보여준다.

```
<timesequence00t-1> <type> <Time>.
<timesequence00t-1> <endTime> <"20000103-1710">.
<timesequence00t-1> <startTime> <"20000103-1709">.
<Kim00a-1> <type> <Person>.
<Kim00a-1> <hasID> <"kim00a-1">.
<Kim00a-1> <hasActivity> <enter>.
<livingRoom00a-1> <type> <LivingRoom>.
<enter> <type> <Activity>.
<enter> <actedInLocation> <livingRoom00a-1>.
<enter> <actedAtTime> <timesequence00t-1>.
<enter> <actedByPerson> <Kim00a-1>.
```

그림 3 트리플 구문으로 표현된 개체 파일

5. 사용자 프로파일을 구축을 위한 사용자 기호 학습

제안하고자 하는 지능형 서비스 에이전트가 사용자에게 적절한 서비스를 제공하기 위해서는 사용자의 기호가 기록되어 있는 사용자 프로파일이 필요하다. CoBrA와 GAIA와 같은 기존의 온톨로지 기반의 유비쿼터스 시스템들은 사용자 프로파일을 각 사용자의 홈페이지에서 가지고 온다. 이러한 방식은 사용자가 자신의 기호를 직접 작성해야 하기 때문에 일정시간에 좋아하는 TV 프로그램과 같이 시간의 흐름에 따라 항상 변하는 기호 정보에 대해서는 그다지 유용하지 않다. 따라서 시스템은 보다 효율적인 서비스를 제공하기 위해 일정 시간(일주일 또는 한달)마다 동적으로 사용자 프로파일을 생성하여 사용할 필요가 있다.

사용자 프로파일 구축에 있어서의 가장 큰 핵심은 유비쿼터스 환경에서 사용자에게 제공되는 서비스들과 그 서비스에 대한 사용자의 반응을 정확하게 모니터하고, 사용자의 기호를 파악하여, 사용자가 선호하는 서비스 정보들을 기록하는 것이다. 따라서 제안하고자 하는 지능형 서비스 에이전트가 사용자 프로파일을 구축하기 위한 방법으로서, 먼저 사용자에게 제공된 서비스 리스트에 대한 모니터 테이블을 구축하고, 두 번째로 귀납적 기계학습 알고리즘을 적용하여 사용자의 기호를 학습한다.

5.1 모니터 테이블 구축

모니터 테이블은 서비스에 대한 여러 가지 속성 및

속성 값들과 제공된 서비스에 대한 사용자의 반응이 기록된다. 모니터 테이블 구축에 있어서 중요한 것은 속성 필드의 선택이다. 보통 사용자 기호 학습을 위한 속성 필드 선택을 특징 추출(Feature Selection) 단계가 필요하다. 그러나 온톨로지는 사용자 기호학습에 필요한 속성들을 Property형태로써 명확하게 기술하기 때문에, 온톨로지를 이용하여 모든 상황 정보를 관리하는 본 시스템에서는 사용자 기호 학습에 필요한 속성 및 속성 값을 정확하게 제공 받을 수 있다.

모니터 테이블의 또 하나의 중요한 구상요소로 제공된 서비스에 대한 사용자의 반응이다. 즉, 제공된 서비스에 대해 사용자가 특정한 반작용(선택해 준 TV 채널 대신 사용자가 다른 채널을 선택)이 있으면, 사용자는 그 서비스에 대해 부정적인(Negative) 반응을 보이고, 특정한 반작용이 없다면, 사용자는 서비스에 대해 긍정적인(Positive) 반응을 보인다고 간주 하는 것이다. 표 3은 사용자 기호 학습을 위해 사용되는 모니터 테이블의 일부다.

5.2 사용자 기호 학습

본 논문에서는 유비쿼터스 환경 내에서 사용자의 기호를 동적으로 학습하기 위해서 엔트로피 개념을 활용하는 C5.0을 이용한다. C5.0은 귀납적 기계학습 알고리즘 중의 하나로서 학습 예제들을 대상으로 대표적인 특성(Feature)을 발견하고 분석한다[15,16]. 따라서 본 논문에서의 학습 예제는 모니터 테이블에 기록된 서비스에 대한 기록이며, 분석된 정보는 특정 영역을 분류하는 모델을 구성한다. 본 논문에서는 두 종류의 모델이 생성된다. 즉 선호 모델과 비 선호 모델이 생성되며, 결국 선호 모델 내에는 사용자의 기호 정보(Name, Location, startPtime, endTtime, Activity, Device, Service)가 담기게 된다.

C5.0은 모델을 만들기 위해서 정보 이론(Information Theory)에 근거하는 Gain을 사용하는데, Gain값을 구하는 식은 다음과 같다[17,18].

$$Gain(S,A)=Entropy(S)-\sum_{v\in Values(A)}\frac{|S_v|}{|S|}Entropy(S_v) \quad (1)$$

여기서 S는 학습 예제의 전체 집합이며, A는 속성을 나타내고, Sv는 속성값을 나타낸다. C5.0을 이용하여 생성되는 모델은 결정 변수를 나타내는 내부 노드와 분류 결과를 나타내는 단말 노드 그리고 이들을 서로 연결하는 선(edge)을 기반으로 생성된다. 따라서 모델로 형성된 사용자의 기호 정보는 3.4절의 지식 베이스에 사실로 저장된 사용자 기호로 표현 예처럼 사용자 프로파일 데이타베이스에 저장되며, 서비스 추론엔진이 적절한 서비스를 추론하는데 사용된다.

6. 실 험

본 논문에서 제안하는 온톨로지 기반의 지능형 서비스 에이전트는 에이전트간의 상황 정보 공유를 통한 메시지 전달과 자동 생성된 사용자 프로파일을 기반으로 서비스 추론을 통해서 사용자에게 적절한 서비스를 제공하게 한다. 이와 같은 기능은 사용자의 행위를 에이전트가 이해할 수 있는 상황 정보로 표현하고, 이러한 상황 정보와 사용자 프로파일의 내용을 기반으로 하는 추론 엔진이 구축되어야 하며, 추론 엔진에서 추론된 서비스가 서비스를 요청한 에이전트에게 정확하게 전달되어야 가능하다. 그래서 본 논문에서 제안하는 시스템의 실행 가능성을 평가하기 위해 에이전트 행위 모듈에 적용되는 추론 엔진과 추론된 서비스의 메시지 통신을 실험해 보았다.

본 실험을 위해 구현된 에이전트 프로토타입은 자바(JAVA) 기반의 에이전트 개발 프레임워크인 JADE를 사용하여 구현되었다. JADE는 그래픽 사용자 인터페이스(Graphic User Interface : GUI)를 제공하여 JADE를 통해 구현된 에이전트의 실행 주기와 에이전트간의 메시지 통신을 통하여 서비스 실행을 직접 확인할 수 있기 때문에, 구현된 지능형 서비스 에이전트가 정확히 동작하는지 확인할 수 있다. 유비쿼터스 온톨로지는 OWL(Web Ontology Language)로 구축하여 온톨로지 서버에 저장하였으며, 상황 정보 데이타베이스로부터 상황 정보를 받아들여 서비스를 추론하는 추론 엔진을 구축하기 위해 JESS(Java Expert System Shell)[11]를 사용하였다. JESS는 썬 마이크로 시스템에서 제작한 규칙-기반 전문가 시스템 저작 도구이며, 자바 환경에서

표 3 모니터 테이블

:NAME	.Location	startPtime	endTtime	Activity	device	Service	Reaction
Kim00a-1	livingRoom00a-1	1700	1800	Stay	Light	Lux_70	Positive
Kim00a-1	livingRoom00a-1	2200	2240	Watch_TV	TV	Ch_7	Positive
Kim00a-1	livingRoom00a-1	0900	0930	Stay	Light	Lux_30	Positive
Kim00a-1	livingRoom00a-1	1200	1210	Out	Light	Lux_00	Positive
Kim00a-1	livingRoom00a-1	1700	1800	Stay	Light	Lux_50	Negative
Kim00a-1	livingRoom00a-1	1400	1420	Enter	Light	Lux_50	Positive

구동된다.

제안하고자 하는 시스템의 실행 가능성을 시험하기 위해 한 개씩의 루트 에이전트, 서비스 액션 큐 에이전트와 6개의 장치 에이전트(텔레비전, 형광등, 매직미러, 오디오, 에어컨, 웹 패드)를 에이전트 플랫폼에 생성하였으며, 발생할 수 있는 사용자의 행위를 6개의 시나리오로 정의하여, 적절한 서비스가 제공되도록 메시지가 전달되었는지 확인해 보았다. 본 실험을 위해 구축된 루트 에이전트는 사용자의 행위가 기록된 온톨로지 개체 파일이 들어오면 서비스 요청 계획 따라 필요한 서비스를 제공하는 에이전트에게 서비스를 요청하는 역할을 하며, 서비스 에이전트는 사용자 기호를 나타내는 사실들을 바탕으로 서비스를 추론하여 루트 에이전트나 서비스 액션 큐에 전송한다. 또한 서비스 액션 큐 에이전트는 전송 받은 서비스 메시지에 따라 각 장치의 상태를 변경하는 역할을 한다. 그림 4는 본 논문에서 제안하는 다중 에이전트 시스템의 전체적인 구조를 나타내고 있다.

표 4는 본 실험에 사용된 사용자 행위 시나리오다. 본 시나리오는 시간의 흐름 순으로 작성되었다.

시나리오에 정의된 Kim의 행위와 Kim의 사용자 프로파일을 사실로 변환하여 서비스 규칙이 적용된 추론 엔진에 적용한 결과 그림 5와 같은 실행 결과를 얻을 수 있었다. 또한 그림 6은 추론된 서비스를 요청한 에이전트에게 정확히 전달이 됐는지를 보여준다. 시나리오의 순서에 따라 각각의 사용자 행위를 입력 받은 루트 에이전트는 ACL 메시지를 통해 필요한 서비스를 제공하는 장치의 에이전트에게 사용자 행위가 표현된 상황 정보와 함께 서비스 요청을 전송한다. 그림 5는 시나리오에 따라 서비스 에이전트들이 Kim의 사용자 프로파일을 기반으로 추론한 서비스의 결과를 차례대로 나타내고 있다. 즉, "(light (status on)(lux 80)(location living))"은 거실의 형광등 에이전트가 "형광등을 켜고, 조도를 80으로 조절"이라는 의미이다. 추론된 서비스는 서비스를 요청한 루트 에이전트와 각 장치를 제어하는 서비스 액션 큐에 메시지로 전달되며, 그림 6은 전체적인 메시지 전달과정을 나타내고 있다. 그림 6은 시나리오 1번의 행위에 따른 서비스 메시지 전달의 예를 보인 것이다. 즉 Kim이 7시에 거실에 들어오면, 루트 에이전트는 형광등 에이전트에게 작동 및 조도 조절 서비스를 요청하며, 서비스를 요청 받은 에이전트는 서비스 액션 큐에게 현재 형광등의 상태를 확인한 후 Kim의 기호에 맞는 형광등 서비스를 추론한다. 형광등 에이전트는 추론된 서비스 내용을 루트 에이전트와 서비스 액션 큐에게 전송하여 사용자에게 적합한 서비스(장치들의 기능 조절)가 실행되도록 한다.

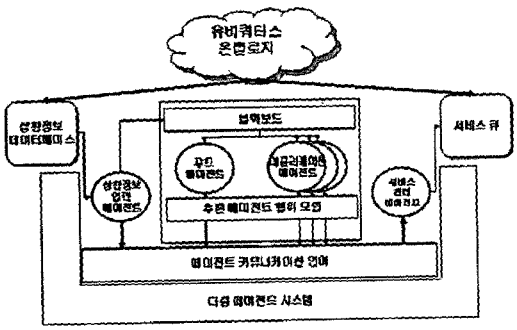

그림 4 다중 에이전트 시스템의 구조

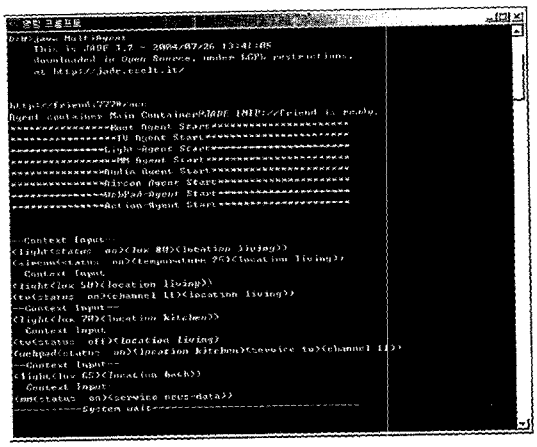

그림 5 입력된 상황 정보에 따라 추론된 서비스 메시지

표 4 사용자 행위 시나리오

시나리오 번호	시나리오	제공될 서비스
1	Kim이 7시에 거실로 들어왔다.	형광등 에이전트 - 형광등 작동, 조도 조절 에어컨 에이전트 - 에어컨 작동, 온도 조절
2	Kim이 7시 10분에 거실에서 TV를 본다.	형광등 에이전트 - 조도 조절 TV 에이전트 - TV 작동, 채널 조정
3	Kim이 7시 15분에 식당에 들어갔다.	형광등 에이전트 - 조도 조절
4	Kim이 7시 20분에 음식을 먹는다.	TV 에이전트 - TV 작동 중지 웹 패드 에이전트 - 웹 패드 작동, 서비스 조정
5	Kim이 8시 25분에 욕실로 들어갔다	형광등 에이전트 - 조도 조절
6	Kim이 8시 30분에 씻는다.	매직미러 - 매직미러 작동, 서비스 조정

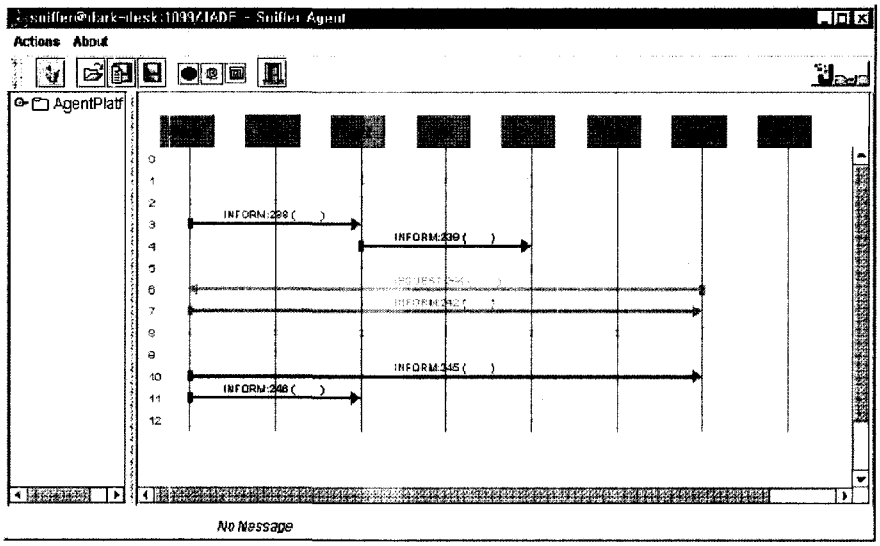

그림 6 서비스 메시지 통신 과정

7. 결 론

본 논문에서는 에이전트간의 상황 정보 공유를 통한 에이전트간의 원활한 메시지 전달과 사용자에게 적절한 서비스를 제공하기 위한 온톨로지 기반의 지능형 서비스 에이전트를 제안하였다. 유비쿼터스 컴퓨팅의 궁극적인 목적은 수많은 지능형 컴퓨터들이 유기적으로 연결되어 언제 어디서나 사물(Object)을 인식하고 사람들에게 필요한 정보나 서비스를 제공하는 것이다. 본 논문에서 제안한 시스템의 목적은 온톨로지를 적용하여 각각의 유비쿼터스 서비스 시스템 내에서의 상황 정보(Context Information)간의 차이를 조절하여 공유가 가능하게 하고, 사용자의 취향을 서비스에 반영하는 것이다. 이러한 특징을 가지도록 하기 위해 유비쿼터스 온톨로지의 적용과 추론엔진이 적용된 에이전트-행위 모듈 구축 및 사용자 프로파일 생성 방법에 대해서 연구를 진행 하였다. 유비쿼터스 온톨로지는 여러 가지 형태로 감지되는 상황 정보의 속성과 구조를 일관성 있는 개념으로 명시하기 때문에, 유비쿼터스 환경 안에 존재하는 각각의 에이전트들이 서로 다른 형태의 상황 정보의 의미를 이해할 수 있게 하고, 귀납적 기계학습의 적용은 시스템 스스로 사용자 프로파일을 구축하게 하며, 이러한 프로파일을 바탕으로 추론 엔진이 적용된 에이전트 행위 모듈은 규칙 베이스 내에 정의된 서비스 규칙을 이용하여 사용자에게 적합한 서비스를 추론하며, 추론된 결과를 ACL 메시지로 만든 후 서비스를 요청한 에이전트에게 전송하기 때문에 시스템으로 하여금 사용자에게 적절한 서비스를 제공할 수 있게 한다. 이러한 시스템의 특징은 유비쿼터스 컴퓨팅의 목적과 부합되기 때문에 제안된 온톨로지 기반의 지능형 서비스 에이전트를 유비쿼터스 컴퓨팅 환경을 지원하는 여러 시스템에 적용 가능하다.

본 시스템에서는 유비쿼터스 환경에 적용할 사용자 프로파일 학습 방법 및 적용에 있어서, 대용량의 학습 예제의 처리와 학습 속도를 고려하지 않았다. 현재 이 점을 해결할 수 있는 연구를 계속 진행 중이며, 향후에는 유비쿼터스 컴퓨팅 환경 내에서의 사용자의 행위와 제공 받은 서비스를 좀더 효과적으로 모니터 하여, 시간과 공간을 고려한 상황 데이타들로 적절하게 클러스터링 하는 방법과 클러스터링 된 상황 데이타들로부터 사용자의 기호를 효율적으로 학습하는 방법을 연구할 것이다.

참 고 문 헌

[1] 김지인, "유비쿼터스 컴퓨팅: 어떻게 할 것인가?", 정보과학회지, 제 21권, 제 5호, pp.5-17, 2003년 5월.
[2] "Fifa ACL Message Structure Specification," http://www.fipa.org/specs/fipa00061G.html
[3] Dey A.K., et al. "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," anchor article of a special issue on Context-Aware Computing, Human-Computer Interaction(HCI) Journal, Vol.16, 2001.
[4] "About Context Broker Architecture," http://cobra.umbc.edu/about.html
[5] Harry Chen, Tim Finin, and Anupam Joshi, "An Ontology for ContextAware Pervasive Computing Environments," In Proceedings of Workshop on

Ontologies and Distributed Systems, in conjunction with IJCAI 2003 Conference, Acapulco, Mexico, August 2003.
[6] Anand Ranganathan, Roy H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In ACM/IFIP/USENIX International Middleware Conference, Rio deJaneiro, Brazil, 2003.
[7] OWL Web Ontology Language Semantics and Abstract Syntax, http://www.w3.org/TR/owl-semantics/, W3C Recommendation 10 February 2004.
[8] FIPA, "Agent Management (TC1)," FIPA '97 Draft Specification, 1997.
[9] 최중민, "시멘틱 웹 개요와 연구동향", 정보과학회지, 2003년 3월.
[10] RDF Primer, http://www.w3.org/TR/rdf-primer, W3C Working Draft 23 January 2003.
[11] Ernest F.H, "Jess In Action," Manning Publications Co, 2003.
[12] Georg F. Luger, "Artificial Intelligence," Pearson Education Limited, pp.93-96, 2002.
[13] "Foundation for Intelligent Physical Agent," http://www.fipa.org/specs/fipa00001/SC00001L.html
[14] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, Giovanni Rimassa, "JADE Programmer's Guide," 21 February 2003.
[15] Tom Mitchell, Robert Armstrong, Dayne Freitag and Thorsten Joachimes, "WebWatcher : A Learning Apprentice for the World Wide Web," 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, Stanford, March, 1995.
[16] T.M.Mitchell, "Machine Learning," McGraw Hill, 1997.
[17] R. Quinlan, "Introduction of Decision Tree," Machine Learning, pp. 81-106, 1986.
[18] JR. Quinlan, "C4.5 Programs for Machine Learning," SanMateo, CA: Morgan, Kaufman, 1993.

김 제 민
정보과학회논문지 : 소프트웨어 및 응용
제 33 권 제 3 호 참조

박 영 택
정보과학회 논문지 : 소프트웨어 및 응용
제 33 권 제 3 호 참조