에이전트 기반 상황 인식 모델 및 응용 현황

가톨릭대학교 김지환* · 양정진**

1. 서 론

근래의 '컴퓨테이션'은 다양한 기기들에 내제되는 기능으로, 작고 가벼운 랩탑이나 노트북들도 PC만큼이나 강력한 처리기능을 가지고 있어 사용자의 처리기능 활용에 이동성의 폭을 넓혀준다. 한편으로는, '컴퓨테이션'은 퍼스널 기기의 개념을 벗어나 벽사이즈의 디스플레이를 통하여 사람들이 서로 소통하고 정보를 접할 수 있게 한다. 이는 물리적으로 공유된 위치에서 사람들 간에 서로 협력적으로 일하는 것을 지원하는 것을 의미하기도 한다.

"퍼베이시브"라는 용어가 Mark Weiser[1]에 의해 처음 소개되면서 이는 이음새 없는 순조로운 기기의 통합을 통하여 기기와 기술적인 요소는 배경으로 하고 실제로 사용자의 편리를 중심으로 사용자와 사용자의 타스크를 주된 쟁점으로 다루는 것을 말한다. 상황인식 시스템은 퍼베이시브 컴퓨팅의 한 분야로, 사용자의 특정한 조정이나 간섭 없이도 시스템의 작동이 현재 상황을 고려하여 이에 적절하게 이루어지며 이러한 환경적인 상황을 고려함으로써 시스템의 유용성과 효율성을 높이고자 한다. 특히, 이동기기를 사용하는 경우에는 작동하는 프로그램과 서비스가 인식되는 기기의 위치, 현재 시간, 다른 환경적 특징들에 적절하게 반응하고 상황적인 데이터가 변함에 따라 이에 적응력 있게 대처하는 것이 요구된다. 이때 필요한 상황적인 정보는 센서적용, 네트워크 정보, 기기 상태, 사용자 파일의 참고등 다양한 방법을 통해 얻어질 수 있다.

예를 들어, 적외선 기반 시스템은 사용자의 위치 파악이 가능하여 전화연결을 사용자에게 근접한 전화기로 전달해 준다든지 하는 서비스를 할 수 있다. 이러한 위치 정보 외에도 다른 종류의 상황정보 활용이 시도되고 있으며 그 종류는 상황에 대한 정의에 따라 조금씩 다르게 규정된다. 상황에 대한 다양한 정의 중에서 대표적인 Dey와 Abowd[2]의 정의는 상황을 '사용자와 애플리케이션 간의 상호작용에 관련되는 엔티티들 즉, 사람, 장소 또는 객체의 상태를 나타내는데 활용되는 모든 정보'라고 규정하고 있다. 상황의 사례를 분류하는 방법 중에 가장 많이 적용되는 것으로는 Perkop과 Burnett[3]에 의한 external과 internal 범위로 구분한 것과 Hofer[4]의 physical과 logical 상황으로 구분한 것이 있다. 외적인 (혹은 물질적인) 상황의 범위는 위치, 빛, 소리, 움직임, 촉감, 온도나 공기압력 등 흔히 하드웨어 센서로 측정 가능한 상황을 일컬으며 반면에 내부적인 (혹은 논리적인) 상황 정보는 사용자의 목표, 타스크, 업무상황, 비즈니스 처리, 또는 사용자의 감정 상태와 같이 사용자에 의해 주어지거나 사용자의 상호작용을 관찰함으로써 얻어지는 것들이다.

현재 대부분의 상황인식 시스템은 위치정보와 같은 외적인 상황을 활용하고 있으며 이러한 외적인 특성들은 바로 적용 가능한 센싱 기술들을 활용하여 쉽게 얻어질 수 있다. 또한 대부분의 연구들은 상황정보를 수집하기 위한 재사용 가능한 센싱 인프라구조 생성에 촛점이 맞추어져 있다. 반면에 유비쿼터스 컴퓨팅에서의 상황정보는 '내 호텔에서 가장 가까운 돌아볼 만한 곳은?'과 같은 서비스 자체가 시간, 위치, 사용자 선호도, 사용자 상황의 복합적 범위를 포함하고 있고 이에 관한 질의나 서비스를 가능하게 하는데 사용되며 이는 컨텍스트 모델링과 기존 정보와 상황정보를 바탕으로 새로운 정보를 유추해 냄으로써 유용한 서비스를 제공할 수 있게 된다.

상황정보를 처리하기 위해서는 엔티티를 크게 3가지 : 장소(방, 빌딩), 사람(개인, 그룹), 사물(실질적 객체, 컴퓨터 컴포넌트)로 분류하여 볼 수 있다[5]. 이들 각각의 엔티티들은 그들이 지닌 특징들에 따라서 신원, 위치(엔티티의 위치, 동일한 위치, 근접함), 상태(or 활동, 고유의 특성- 온도, 방의 조명도, 기기에서 현재 작동중인 프로세스), 시간(상황과 이벤트를 나타내는 timestamp)으로 구분된다.

이러한 환경은 서비스를 사용하는 사람과 이들이 사

* 학생회원
** 정 회 원

용하는 디바이스들이 이동성이 있으면서 각기 Global System for Communications(GSM), General Packet Radio Service(GPRS), Universal Mobile Telecommunications System(UMTS), Wireless LAN(WLAN), Bluetooth 등과 같은 서로 다른 네트워크 기술을 사용하여 통신을 하고 있어 다양한 하드웨어와 소프트웨어의 구성 뿐 아니라 다분히 동적인 성향이 강조되고 있다.

유비쿼터스 환경에서의 애플리케이션들은 자율적이고 지능적으로 업무를 수행하고 목적에 부합되는 상호규약(Protocol)을 이용하여 상호 작용한다. **에이전트는 자율적이고 지능적인 특성을 가지는 애플리케이션이며[6] 에이전트의 물리적인 구성과 상호작용을 위한 표준을 제시하는 Foundation for Intelligent Physical Agents 에 준한[7] 에이전트 시스템들은 상호작용 표준을 적용하여 통신함으로써 상호운용성을 향상**시키고 있다. 본 고에서는 2장에서 상황인식 시스템의 아키텍쳐와 컨텍스트 모델등 상황인식 시스템 디자인의 일반적인 원칙을 조사하고 3장에서는 사용자에게 서비스 제공을 용이하게 할 수 있는 상황인식 소프트웨어 에이전트 개발의 지원 기술들을 살펴본다. 4장에서는 상황 인식 소프트웨어 에이전트 시스템 사례를 살펴보고 5장에서 상황인식 에이전트 시스템의 정형화된 개발방법론의 필요성을 기술하며 결론을 맺는다.

2. 상황 인식 시스템

2.1 Client 기기

유비쿼터스 환경에서 제공되는 서비스는 다양한 client 기기를 통하여 활용될 수 있다. Client 기기들로는 핸드폰에서 랩탑 컴퓨터에 이르는 다양한 종류가 이에 해당되고 이들 기기들은 처리의 기능면에서 많은 차이를 나타낸다. 기기의 수용능력과 제한요소는 사용자가 서비스를 사용하는 방법과 사용자에게 제공되는 서비스 종류를 정하는데 중요한 요소로 작용한다. 수용능력은 처리능력과 인터페이스 성능으로 구분되어지고, 처리능력의 지수는 서비스를 제공하기 위해 client 기기가 어떠한 자원을 지니고 있는지를 정한다. 어떠한 기기는 웹 브라우저와 같이 활용 가능한 서비스로 인터페이스 기능만 가능한 것이 있고, 처리능력이 있는 다른 기기들은 부분 혹은 전체적인 서비스를 수행하여 제공하기도 한다. 인터페이스 성능은 사용자에게 제공되는 서비스의 출력표시의 특징들을 규정하게 된다.

기기의 수용능력은 기기 프로파일로 명시되어 진다. 기기 프로파일에는 기기의 타입, 출력 및 입력 기기와 같은 세부사항이 포함되어 있다. 프로파일 형태의 한 예로 W3C의 Composite Capability/Preference Profile (CC/PP)를 들 수 있다[8]. CC/PP를 사용하여 제공하는 서비스와 함께 각 기기의 프로파일을 등록할 수 있다. 프로파일 등록의 장점은 서비스에 제공되는 컨텐츠가 등록된 프로파일의 정보를 적용하여 화면에 맞게 조정되거나 이미지의 색조 조절 등과 같이 기기의 요구조건에 맞게 조절되어 제공될 수 있다는 것이다.

2.2 상황인식 시스템 아키텍쳐

상황인식 시스템은 다양한 방법으로 디자인될 수 있다. 디자인 방식은 요구되는 사항과 조건에 영향을 받으며 센서의 위치, 사용자의 수, 사용기기에 따른 자원 가용성, 시스템 확장의 용이성등이 이들 요구사항이나 조건에 해당한다. 상황인식 시스템을 디자인하는데 가장 주요한 요소로 작용하는 상황정보 획득방식은 크게 3가지로 나누어 볼 수 있다[9].

- **직접적인 센서 활용** : 센서장치가 장착된 기기에서 사용되는 방식으로 클라이언트 소프트웨어가 필요한 상황정보를 장착된 센서를 통해 직접 수집하는 방식이다. 센서데이터를 모으거나 처리하는 부차적인 계층이 없이 애플리케이션에 기기를 위한 드라이버를 직접 연결하여 사용하는 방식으로 드물게 활용되며 동시에 작동하는 다수의 센서 접근 관리 능력이 부족하여 분산형 시스템에는 부적절하다.
- **미들웨어 인프라구조** : 최근의 소프트웨어 디자인이 비즈니스 로직과 GUI를 분리된 형태로 캡슐화시키는 것과 같이 상황인식 시스템에서 미들웨어 기반 상황정보 획득 방식은 저층의 센싱 관련 세부사항을 숨기고 분리하여 계층적인 아키텍쳐를 활용한다.
- **컨텍스트 서버** : 원격의 데이터 소스를 다수의 클라이언트가 접근하도록 허용하는 방식이다. 분산된 접근은 원격 컴포넌트 관리를 소개함으로써 미들웨어 아키텍쳐를 확장한다. 여기에서 모아진 센서 데이터는 동시에 다중 접속을 용이하게 하기 위해 컨텍스트 서버로 이동하게 된다. 센서의 재활용 외에도 컨텍스트 서버는 자원을 집중적으로 사용하는 클라이언트들에 적합한 장점이 있다. 클라이언트-서버 아키텍쳐를 기반으로 한 상황인식 시스템을 설계할 때 상황인식 시스템에 사용된 단말 장치의 대부분은 처리능력과 디스크 공간 등의 제한을 갖는 모바일 장치이므로 이러한 것은 중요한 고려요소가 된다. 적절한 프로토콜, 네트워크 성능, Quality of Service 파라미터 등도 고려되

어야 하는 대상들이다.

이와 유사하게 Winograd[10]은 다중의 프로세스와 컴포넌트를 통합할 수 있는 세 개의 다른 상황 관리 모델을 제시하였다.

- **Widget** : Widget은 하드웨어 센서를 위한 공통의 인터페이스를 제공하는 소프트웨어 컴포넌트이다. Widget의 캡슐화로 재사용을 통한 응용 개발이 쉬운 편이다. 비슷한 종류의 컨텍스트 데이터는 이를 제공하는 widget을 교환함으로써 이루어진다. 예를 들어, 위치 정보는 라디오 주파수 또는 카메라 widget을 교환하여 얻어질 수 있다. widget 관리자에 의해 통제되는 widget 방식은 강한 결합으로 인해 효율성은 높지만 컴포넌트의 부분적인 오류를 다루는 데에는 적합하지 않다.
- **Networked Service** : 컨텍스트 서버와 유사하지만 조금 더 유연한 방식으로서 네트워크 기반 서비스를 들 수 있다. 컨텍스트 서버의 글로벌 widget 관리자 대신 서비스 발견 기술을 활용하여 네트워크에서 제공되는 서비스를 찾아준다. 이러한 서비스 기반 방식은 복잡한 네트워크 기반 컴포넌트들로 구성되어 있어 widget 아키텍쳐 만큼 효율적이지는 못하지만 견고성을 제공한다.
- **Blackboard Model** : 처리중심인 widget이나 서비스 기반의 모델과는 달리 블랙보드 모델은 데이터중심이다. 블랙보드 모델방식은 블랙보드라는 공유된 매개체에 메시지를 올려놓고 관련있는 특정한 이벤트 발생시 통지를 받도록 신청을 한다. 이 방식은 새로운 컨텍스트의 소스 추가가 간단하고 이들의 형상관리가 용이하다는 장점이 있는 반면, 블랙보드를 운영하기 위한 중앙 집중적 서버가 필요하고 항시 블랙보드를 거쳐 통신이 이루어지기 때문에 통신상의 비효율성이 생기는 단점이 있다.

본고에서는 분산 시스템 내에서 유용하다고 판단되는 미들웨어 기반과 컨텍스트-서버 아키텍쳐에 초점을 맞추었다. 많은 계층적 상황인식 시스템과 framework들이 발전되어 왔고 대부분 기능적인 범위와 계층의 이름은 다르지만 에이전트나 다른 아키텍쳐를 선택적으로 사용하고 있다.

2.3 컨텍스트 모델

컨텍스트 데이터를 기계가 처리할 수 있는 형태로 정의하고 저장하기 위해서는 컨텍스트 모델이 필요하다. 컨텍스트 정보의 표현과 교환방식에 따른 데이터 구조를 기반으로 하여 Strang과 Linnhoff Popien[11]에 의해 정리된 컨텍스트 모델링의 기법들로는 Key-Value 모델, 마크업 스키마 모델, 그래픽 모델, 객체 지향 모델, 로직 기반 모델, 온톨로지 기반 모델 등이 있다.

이들 중 온톨로지 기반 모델링 방식은 단순성, 유연성, 확장성, 일반성, 표현력에 있어 우월함을 나타내어 컨텍스트 모델링에 가장 적합한 것으로 알려져 최근 넓은 범위의 컨텍스트를 다루기 위한 온톨로지 구성이 주요하게 다루어지고 있다. 온톨로지 작성을 지원하는 도구들을 활용하여 RDF, OWL등의 온톨로지 표준 언어로 선언적인 표현들을 나타내게 된다. 하나의 컨텍스트는 주로 컨텍스트의 타입과 이를 센싱하여 얻게 되는 값으로 구성하게 되며 부가적으로 컨텍스트 데이터가 센싱된 시간(timestamp), 정보가 확보된 분야(source), 데이터의 확신성(confidence) 등을 적용하여 상황인식 시스템을 가동하면 더욱 유용하다.

3. 지원 기술(Supporting Techniques)

유비쿼터스 사회에서 추구하는 유용하고 지능적이며 상황-인지적인 서비스를 실현하기 위해서는 다수의 주요 기술들이 요구된다. 유비쿼터스 컴퓨팅 환경에서 일반적인 용어로 사용되는 미들웨어는 상황 인지적 서비스를 개발하는 애플리케이션 개발자들을 위한 소프트웨어 서비스들의 집합들을 의미하고 제공하는 서비스로는 메시징과 커뮤니케이션, 자원 발견, 처리와 저장 서비스 등을 들 수 있다.

지능형 디지털 홈과 같은 지역단위 유비쿼터스 환경은 개인화된 서비스 지원을 위한 사용자 지원환경과 전체적인 지역을 관장하는 소프트웨어 플랫폼 기술로 구분지어 볼 수 있다. 개인화된 서비스를 제공하기 위해서는 개인화에 필요한 사용자 정보의 정적인 특성(이름, 주민등록번호 등)과 현재 사용자의 상태 (위치, 기호도, 감정 등)를 나타내는 동적인 특징을 모델링 하는 것이 필요하며 사용자가 처한 현재의 상황에 따라 개인화된 서비스가 제공될 수 있다. 사용자 지원 환경은 이를 지원하는 소프트웨어 프레임워크 기반에서 이루어진다. 미들웨어 단에서 소프트웨어 프레임워크가 제공하는 일반적인 기능에는 상황관리 기술, 이벤트 전달을 위한 이벤트 서비스 기술, 브로커 서비스 기술 등이 있다.

3.1 에이전트 기술의 적절성

에이전트 기반 상황인식 시스템은 유비쿼터스 컴퓨팅 환경에서 다수의 에이전트들로 이루어져 이들 간의

협력을 통하여 작업을 수행해 나간다. 에이전트는 유비쿼터스 환경에서 효율적으로 동작하기 위해서 현재 상황에 대한 감지(sensing)와 추론(reasoning)능력을 활용하며 다른 에이전트와 원활한 상호작용을 한다.

상황인식 컴퓨팅에서 상황의 주요한 면들은 사용자의 위치, 관련자, 사용가능한 자원들을 의미하며 상황은 때때로 지식베이스를 관리 가능한 집합체로 나누는 수단이 되거나 추론과정을 용이하도록 하는 논리적 생성자 역할을 하기도 한다. 상황은 좀 더 일반적인 의미로 하나의 엔티티가 처해 있는 환경을 파악하는데 사용되는 주요 정보로 이에 해당되는 엔티티에는 사용자와 애플리케이션 간의 상호작용에 관련되는 사람, 위치, 객체들을 들 수 있다. 에이전트 시스템은 이러한 사용자와 에이전트화 된 애플리케이션 구성요소들 간의 상호작용에 기반을 둔 조직 구성에 적합한 기술로서, 상황인식 시스템의 가장 최근의 정의에 의해서도 상황인식 시스템을 모델링하고 개발하는데 가장 적합한 적용기술로 간주된다.

상황 인식을 온전히 실행하기 위해서는 관련 모바일 애플리케이션이 상황정보와 함께 고려되어야 한다. 예를 들어, 유비쿼터스적인 서비스로 주식 트레이딩 서비스를 고려해 보자. 모바일 사용자가 Wi-Fi hotspot에 연결된 랩탑과 Bluetooth에 연결된 PDA, Wireless Application Protocol(WAP) 전화기와 간단한 SMS 기반 통신을 하기위한 GSM 전화기를 사용하는 경우에, 인터페이스와 트레이딩 서비스정보의 컨텐츠는 사용 가능한 터미널과 연결성 기술에 맞추어 조절이 되어야 하고 지역적으로는 자원공유 차원에서 우대 사용자와 보통 사용자를 구별하여 서비스를 제공하는 것이 비즈니스 차원에서 합리적이라고 하겠다. 즉, 사용자가 GSM 전화기를 사용하고 있는 경우, 컨텍스트는 정해진 네트워크 인프라에서 돌아가는 게이트웨이로만 구성이 되고 HTML페이지를 텍스트의 내용으로만 요약하여 SMS메세지로 송신되도록 한다. 또한, 현재 트레이딩 서비스로의 접속이 밀집현상을 보일 때 상대적으로 보통 사용자의 서비스 접속을 제어하는 것이 필요하기도 하다. 이와 같이 높은 이질성과 컴퓨팅 환경의 동적인 요소, 그리고 자원의 부족내지 비지속성은 상황적인 서비스 제공에 결정적인 요소들이다. 이러한 요소들로 복잡성이 높아진 상황적인 모바일 서비스를 제공하기 위해서는 상황정보의 다각적 인식뿐만 아니라 개발을 용이하게 하고 런타임 지원을 가능하도록 하는 미들웨어 솔루션이 필요하다. 모바일 에이전트를 비롯한 에이전트 기술은 이동성, 비동기성, 분산형, 위치 인식면에 에이전트 특징을 강조하고 있어서 참신한 상황인식 미들웨어를 구축하는데 적합한 기술로 인식된다.

3.2 에이전트 시스템 지원 도구 및 기술

유비쿼터스 컴퓨팅 환경은 서로 다른 기기와 비동기적인 상황에서도 합리적이고 연속적이며 상황을 인지할 수 있는 서비스를 제공할 수 있어야 한다. 소프트웨어 에이전트 기술은 비동기식 메시징 기반으로 자치적인 컴포넌트들을 통합하고 이들 간의 협력적인 상호작용을 지원하는 기술로 에이전트는 정의에서부터 분산과 이질적이고 동적인 환경에 적합한 기술로 규정되고 있어 유비쿼터스 컴퓨팅에 부합되는 기술로서 인식되고 있다. 에이전트들은 상위레벨의 대화나 자원과 서비스들 간의 협상에도 적용될 수 있다.

- **에이전트** : 에이전트는 내부 상태들과 행동패턴들을 가지며 외부 환경과 상호작용하여 설계된 목적에 알맞은 내부 상태에 도달하기 위한 유연하고 자율적인 행동을 수행하는 캡슐화된 컴퓨터 시스템이다. 에이전트는 주어진 목적을 달성하기 위하여 다른 에이전트와 상호작용 할 수 있으며 의미적 상호운용성을 보장하기 위한 수단으로 온톨로지와 표준 통신 프로토콜을 사용한다.
- **에이전트 언어** : 에이전트기반의 소프트웨어 엔지니어링을 위하여 에이전트는 정형화된 개념을 이용하여 개발 되어야 한다. 이를 위한 에이전트 언어는 크게 에이전트 기술언어와 에이전트 개발언어로 구분된다.
  에이전트 기술 언어(e.g. PLACA)는 에이전트의 속성과 행동타입을 기술하는데 사용된다. 주로 에이전트의 기능, 목적, 계획, 지식, 믿음, 의도 등의 정보로 에이전트를 기술하게 된다. 에이전트 개발 언어는 수행 가능한 에이전트를 프로그램하기 위한 언어로써 목적된 환경에 적합한 프로그래밍 언어로 구현된다.
- **JADE : FIPA-Compliant** : FIPA는 The Foundation for Intelligent Physical Agents의 약자로써 에이전트 기반기술과 상호 운용성을 위한 표준을 제정하기 위한 단체로 에이전트 플랫폼 모델을 제정하였다. FIPA의 에이전트 플랫폼 모델에 따르면 에이전트 플랫폼에는 기본적으로 에이전트, AMS(Agent Management System), DF(Directory Facilitator), Message Transport System 또는 ACC(Agent Communication Channel)가 존재한다.

AMS는 에이전트 플랫폼으로의 접속과 사용을 관리하는 에이전트이다. AMS는 에이전트 플랫폼 안에 오

직 하나가 존재하며, 에이전트의 상태를 유지하고, 생명주기를 관리한다. DF는 다른 에이전트에게 Yellow Page 기능을 제공하여 준다. 에이전트들은 자신들의 서비스를 DF에 등록하거나, 다른 에이전트로부터 제공받을 서비스를 찾기 위해 DF에게 질의를 한다. Message Transport System은 에이전트 플랫폼 상의 에이전트 간의, 또는 다른 에이전트 플랫폼 상의 에이전트들과의 메시지 전달을 위해 쓰인다.

JADE[12]는 Java Agent Development Framework로써 FIPA-compliant한 멀티 에이전트 시스템과 응용 프로그램의 개발을 도와주는 미들웨어이다. JADE는 에이전트 플랫폼과 에이전트 개발을 위한 패키지의 두 부분으로 나누어진다. JADE는 FIPA가 제정한 표준을 지원하며, 에이전트 개발을 위한 클래스들을 제공함으로써 에이전트 개발을 도우며, 플랫폼 안의 에이전트들을 관리하고 모니터링 하는 그래픽 도구를 제공해준다.

- **경량형 에이전트 플랫폼** - JADE-LEAP : 유비쿼터스 환경에서 에이전트는 일반적인 호스트뿐만 아니라 핸드폰이나 모바일 기기 등의 경량형 디바이스에서도 동작이 가능해야 한다. 이를 위해 JADE-LEAP은 JADE가 PC 또는 서버에서부터 핸드폰이나 PDA와 같은 경량형 기기까지 동작할 수 있도록 하는 에이전트 플랫폼이다.

JADE-LEAP은 세 가지 버전으로 컴파일 되는데, J2SE, PJAVA, MIDP 버전이 있다. J2SE 버전은 PC나 서버에서 동작하는 J2SE 버전이다. PJAVA 버전은 PersonalJava를 지원하는, PDA와 같은 hand-held 기기에서 동작한다. MIDP 버전은 MIDP1.0을 지원하는 핸드폰과 같은 기기에서 동작한다. 즉, 지원하는 Java의 버전에 따라 변형시켜, 각각의 기기에서 동작할 수 있도록 한다. 그림 1은 핸드폰 상에서 동작하는 JADE-LEAP의 MIDP 버전을 보여준다.

3.3 모바일 에이전트 기반 상황인식 미들웨어

유비쿼터스 컴퓨팅에서 환경에 적합한 상황인지적 서비스를 지원하기 위해서는 기존의 분산 시스템에 runtime 의사결정이 가능하도록 이동성의 특징들이 고려된 미들웨어가 필요하다. 이동적인 컴퓨팅은 첫 번째로 사용자의 이동패턴과 서비스 및 사용자 요구사항 변화등 변동성(dynamicity) 에 따라 새로운 컴포넌트나 프로토콜의 확장과 변화를 지원하는 인프라가 필요하다. 두 번째로, 서비스 기능에 대한 요청과 실행에 있어서 사용자와 터미널간의 비동기적인(asynchronicity) 지원이 필요하다. 예를 들어, 네트워크 상에서 사용가능한 대역폭이나 통신의 신뢰성 보장에 제한적인 요소를 가진 무선기기를 사용하므로 연결시간을 최소로 하여 처리하는 것이 요구된다. 또한 특정 애플리케이션의 최적화를 수행하거나 가용 가능한 지역자원을 활용하기 위해서는 애플리케이션 개발자들이 이러한 정보를 확인할 수 있는 기능이 제공되어야 한다.

모바일 에이전트(MA) 기반 미들웨어는 사용자(user), 사용하는 터미널(access terminal), 필요한 자원(resource)과 서비스 컴포넌트와 관련된 nomadic/roaming 이동성을 고려한 모바일 컴퓨팅 미들웨어에 가장 적합한 기술로 여겨진다. MA 미들웨어는 이동과 관련된 변동성(dynamicity)에 대해 코드의 동적인 분산/수정과 동적인 자원 바인딩을 에이전트의 상태정보와 함께 이동 가능하도록 지원한다. MA 패러다임은 MA를 모바일 기기에서 사용하는 네트워크 인프라에 투입시키는 용도로 네트워크로의 연결성을 최소화함으로써 비동기성(asynchronicity)을 지원한다. 또한, MA는 그 자치성을 활용하여 고정된 네트워크에 이동적인 proxy 역할을 하거나 사용자 선호도나 사용하는 기기의 프로파일에 따라 서비스를 절충하여 제공하는 역할을 하기도 한다. MA 기술은 상대적으로 보안(security)이나 상호운용성(interoperability)에 약

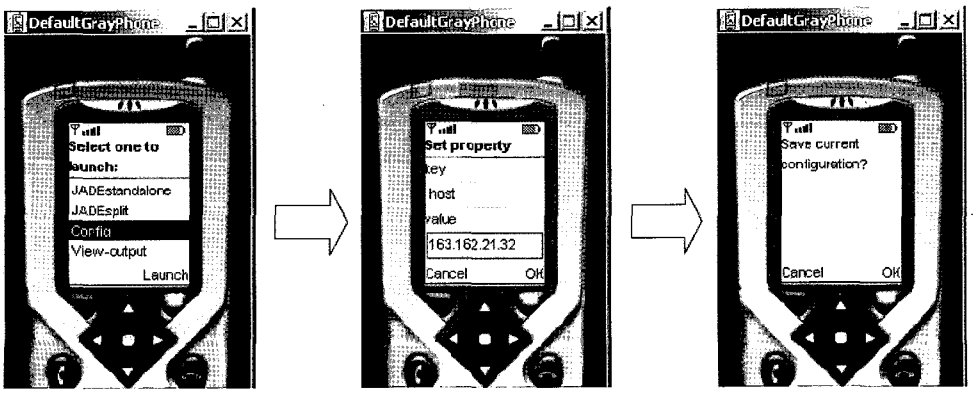

그림 2 Setting properties through the Config MIDlet

한 성향을 지니기도 하였으나 보안 문제는 MA간 통신과 자원 사용권한에 public key 인프라를 활용하고 혹은 모바일 기기에 권한을 적용함으로 완화시키며, 사용 가능한 자원과 서비스 간의 상호작용은 CORBA, MASIF, FIPA와 같은 상호운용적이고 표준화된 인터페이스를 사용하여 상호운용성을 향상시키고 있다.

상황인식을 위한 모바일 에이전트 기반 미들웨어는 상황정보에 대한 센싱, 처리, 제어, 관리 기능을 제공하고 모바일 애플리케이션에 가장 최근에 입수된 상황정보를 적용하게 함으로써 상황적인 모바일 서비스의 디자인과 구현을 단순화 시키는 역할을 담당한다.

MA 기반 미들웨어가 해결해야 하는 대표적인 기능을 보면:

- **이질적 실행 환경에서의 상황정보의 가시도(visibility)와 추출 방법**

구체적으로, 상황인식 시스템에서의 다양성은 데이터와 관련된 실행 환경으로 네트워크 연결성, 통신의 대역폭, 업무 처리량, 사용가능한 지역 정보가 이에 해당하고, 사용자관련 정보로는 사용자 선호도 프로파일, 사용자 주변의 존재를 들 수 있으며, 물리적인 환경의 조건으로 소음의 정도, 온도 등이 있다. MA 시스템은 이러한 정보들의 가시성을 때때로 자바 환경의 이식성을 활용하여 또는 플랫폼에 따라 다른 이질적인 센싱 기술(multi-channel sensing)의 통합을 통하여 얻는다. MA 시스템의 visibility 확보의 예로서, Advanced Mobile Application Support Environment(AMASE)[13]는 GPS를 통하여 터미널의 위치를, Secure and Open Mobile Agents(SOMA)[14]는 WiFi/Bluetooth에 연결된 SNMP 에이전트를 통해 특정 roaming의 셀(cell)기반 위치를 확보한다.

- **서비스 세션 내에 모바일 클라이언트의 컨텍스트를 결정하는 처리 및 합성방법**

상황정보의 이질성과 센싱 매커니즘의 다양성으로 인해 상황정보 미들웨어는 이들을 처리하고 통합하는 기능을 제공해야 하는데 MA 미들웨어는 저층의 모니터링 매카니즘과 상위 계층의 상황인식 MA를 분리함으로써 MA의 개발을 쉽게 하고 컴포넌트 재사용의 가능성을 향상시킨다. CoolAgents[15]는 widget 을 활용하여 이질적인 상황정보를 수집하고 이를 센싱 매카니즘과 관련 없이 동일한 형태로 제공한다.

- **적응적인 서비스를 제공하기 위해서 상황 인식의 유연한 방법**

상황에 따른 서비스 관리로 policy 기반과 reflection 기반이 있다. MA미들웨어에서 policy 기반은 이벤트 상황정보의 변동에 따라 관련 policy를 적용하고 reflection 기반은 상황에 따른 서비스 로직을 기본 레벨과 메타 레벨로 분리하여 상황의 가시성/처리에 따라 또는 서비스 변동을 위한 메타객체를 활용하여 상황에 대처하는 방식이다.

- **서비스의 질 (QoS)적인 요소를 고려하여 수집된 상황의 visibility 를 활용하는 방법**

특히 신뢰성이 보장되지 않는 best-effort 네트워크나 잦은 클라이언트의 이동이 있는 경우 서비스 세션이 이루어지는 동안 QoS 수준에 맞추어 서비스를 제공하는 것은 매우 중요하다. SOMA는 QoS 관리와 모니터링 기능을 제공하고 MONADS는 예전의 상황과 변화과정을 통하여 상황정보의 예측기능을 제공한다.

MA 시스템의 사례들을 통하여 앞으로 해결해야 하는 문제들로 적절한 일반성을 지닌 자원 바인딩과 이동성에 대한 방안의 표현방법과 동적이고 오픈 서비스 조합에 대한 지원 방안을 들 수 있다. 최근 XML 기반의 표준화된 웹 서비스는 오픈 서비스 조합의 한 방법이라고 할 수 있다.

4. 상황인식 소프트웨어 에이전트 시스템

사람은 각각 다른 상황에서 다르게 행동한다. 그들은 그들이 처한 환경에 대해 감지하여, 현재 상황에 적합한 행동을 한다. 사람이 자신을 맞춰가는 행동의 방식은 그들의 경험에 의해 습득한 규칙(Rule)에 의해서이다. 지능형 에이전트들도 그들의 동작하는 상황에 대해 감지하고 추론할 수 있다면 사람과 같은 방식으로 현재 상황에 적합한 행동을 할 수 있을 것이다. 상황인식 소프트웨어 에이전트를 실현시키는 방법은 사람이 상황인식을 하는 것과 유사하게 온톨로지의 공유, 상황을 센싱하는 작업 그리고 이미 알고 있는 것을 토대로 한 현재상황의 추론이 절대적이라고 하겠다[16].이번 장에서는 상황인지에 의한 멀티 에이전트 시스템의 적용 사례들을 알아본다.

상황인식 애플리케이션 개발을 단순화하기 위해서는 추상적인 프레임워크가 필요하다. 상황(Context)의 형태는 물리적 상황(시간, 장소), 환경적 상황(날씨, 조명, 소리), 정보 상황(주식 정보, 스포츠 스코어), 개인적 상황(건강, 기분, 스케줄, 활동), 응용 프로그램 상황(수신된 이메일, 방문한 사이트)과 시스템 상황(네트워크 트래픽, 프린터의 상황)등 여러 종류가 있다. 에이전트들은 그들의 행동에 적합한 상황(Context)을 획득하고 추론할 수 있어야 한다. 상황인지를 위한 미들웨어가 제공해야 하는 서비스는 표 1과 같다.

표 1 상황인지를 위해 미들웨어가 제공해야 하는 서비스

• 다른 센서들로부터의 상황정보의 수집과 다른 에이전트들로의 상황정보의 전달을 지원
• 낮은 레벨 센싱된 상황으로부터 높은 레벨의 추론된 상황까지 지원
• 에이전트의 추론 및 러닝 메커니즘의 사용 지원
• 에이전트에게 다른 상황 하에서 다른 행동들을 하게하는 능력
• 서로 다른 에이전트들 간의 구문과 의미적 내부 소통 지원

Architecture

가장 일반적인 디자인은 하나 이상의 중앙 집중된 컴포넌트들로 이루어진 전통적인 계층구조이다. 이 방식은 자원이 제한적인 모바일 기기의 메모리, 프로세스의 제약을 극복할 수 있게 하지만 견고성이 부족한 면이 있다.

〔SOCAM: Service-Oriented Context-Aware Middleware〕 Gu〔16〕등에 의해 개발된 SOCAM은 상황인식 모바일 서비스의 구축과 빠른 프로토타이핑을 제공한다. 상황인식 모바일 서비스는 아키텍쳐의 최상위에 위치하여 각 서비스는 서로 다른 상황정보를 활용하고 이에 따라 적응력 있게 에이전트의 behavior를 취한다.

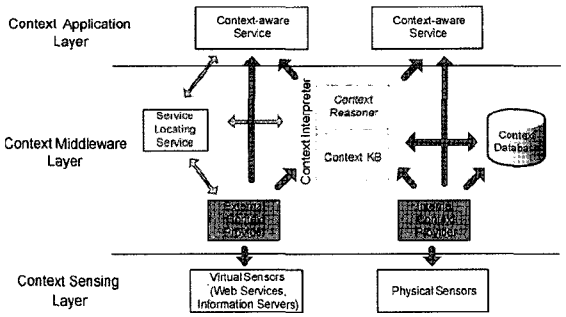

그림 3 SOCAM Architecture

〔CoBrA: Context Broker Architecture〕 Chen〔18〕 등에 의해 개발된 CoBra는 에이전트 기반의 지능형 공간에서 상황인식을 제공하기 위한 아키텍쳐를 갖고 있다. CoBrA 시스템에서 지능형 공간은 물리적 공간으로 지능형 시스템이 공간에 스며들어 사용자에게 서비스를 제공한다. 지능형 context broker는 에이전트 커뮤니티 사이에서 context model을 공유, 관리한다. 각 에이전트는 서용자가 휴대하는 모바일 기기에서 동작하는 응용일 수 있고, 각 상태 정보를 웹상에서 제공하는 웹서비스일 수도 있다. Broker는 Context KB, Context inference engine, context acquisition module, privacy management module로 이루어져 있다. 병목현상을 피하기 위한 broker 연합 생성 기능도 포함하고 있다.

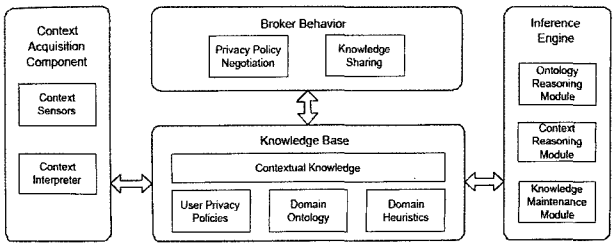

그림 4 CoBrA Architecture

〔CoolAgent RS〕 CoolAgent RS는 FIFA-compliant JADE agent를 기반으로 상황정보의 표현과 분석을 분리시키고 표현되고 공유되고 조작되어진 상황정보를 허락하는 지식 표현 기반구조를 가진다. 또 협업적인 추론 메카니즘을 제공한다. CoolAgent RS는 온톨로지 멥핑서비스를 제공하는 notification agent proxy와 entity tracking agent, venue agent, personal agent, meeting agent, document recommendation agent, food service agent로 이루어져 있다.

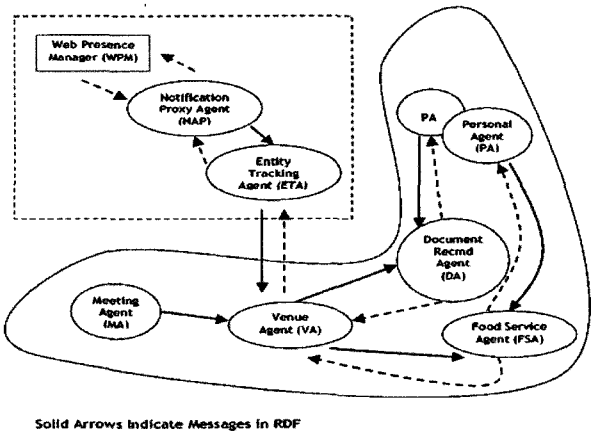

그림 5 CoolAgent RS Architecture

〔Gaia project〕 Roman〔19〕 등에 의해 개발된 Gaia 는 일반적인 운영체제 개념에 상황인식을 포함하여 확장된 형태이다. 서비스를 쿼리로 추출하고 현존하는 자원을 활용하여 현재 상황을 접근해 사용한다. 사용자 중심의 자원을 인식하고 멀티 디바이스를 지원하는 상황에 민감한 모바일 애플리케이션 개발을 위한 프레임워크를 제공한다. Gaia는 물리적 환경을 둘러싸는 유비쿼터스 컴퓨팅 환경인 Smart Space를 위한 기반을 제공해 준다. Gaia는 물리적 공간(Physical Space)과 물리적 공간의 유비쿼터스 컴퓨팅 장치들을 프로그램 가능한 컴퓨팅 시스템으로 변환시켜준다.

Active Space는 물리적 공간(Physical Space)과 사람과의 상호작용을 위한 원활하게 하는 공간이다. Gaia는 event manager, context service, context file system, component repository, presence service, space repository로 이루어져 있다. 그림 5는 Gaia의 구조도이다.

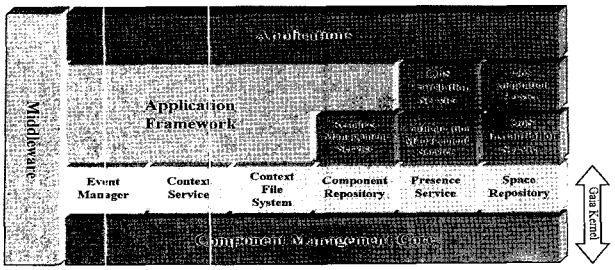

그림 6 GAIA Architecture

Gaia 미들웨어는 에이전트가 여러 종류의 상황 정보의 획득과 추론을 위한 여러 방법을 제공한다. 상황인지는 다음과 같은 원리로 이루어지며 여기서 Context Provider는 여러 상황을 감지하고 다른 에이전트로부터의 접속을 허용하며 다른 에이전트로부터의 질의에 대한 인터페이스를 제공하고, 현재 상황에 대한 질의에 응답한다. Context Synthesizer는 Context Provider로부터 감지된 상황을 전달 받고 추론을 통해 더 높은 단계의 Context를 생성해 낸다. Context Consumer는 상황에 대한 소비를 하는 에이전트로서 Context Provider에게 질의를 하고 이에 대한 응답을 받아들인다. Context Provider Lookup Service는 Context Consumer가 다른 종류의 상황에 대해 질의 할 때, 알맞은 상황정보를 제공하는 Context Provider를 검색하도록 하는 역할을 한다. Context History는 현재 상황이 저장되어지는 데이터베이스이다. 수집된 상황정보는 다양한 데이터마이닝 기법을 사용하여 사용자 행동에 대한 패턴을 알아내는 데 사용되어진다. Ontology Server는 에이전트간의 원활한 상호운용성을 위한 온톨로지가 저장되어진다.

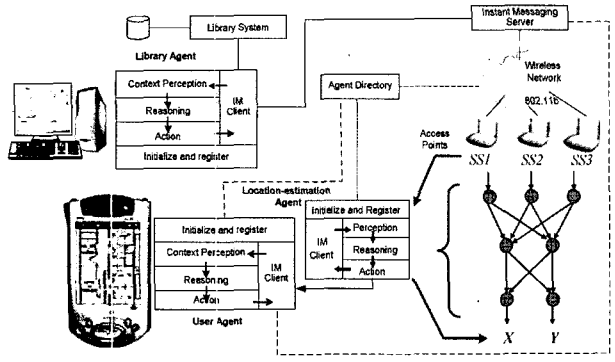

그림 7 SALSA Architecture

[SALSA]는 명명, 등록, 인증 그리고 에이전트의 위치 찾기 그리고 이런 특성을 가지는 자율적인 에이전트를 개발하기 위한 간단한 API를 제공한다. SALSA는 모바일 사용자를 위한 모드를 제공하고 사용자와 기기사이의 상호운용성을 위해 현재 존재하거나 새로운 서비스사이의 상호작용 메커니즘을 가진다. SALSA[20]의 구조는 그림 7과 같다.

Resource Discovery

분산 네트워크 상의 센서는 언제든지 고장이 나거나 추가될 수 있는 동적인 요소가 존재하기 때문에 실행시간에 활용 가능한 적절한 센서를 발견하는 일은 중요하다. SOCAM은 service locating service를 이용한 서비스 발견 메커니즘을 제공한다. Gaia는 각기 다른 context provider가 등록 컴포넌트에 적재된다. Context Toolkit [21]은 에이전트 지향 시스템은 아니지만 발견자를 제공한다. 등록 컴포넌트를 활용하여 interpreter, aggregator, Widget등은 등록을 하게 되어 있고 발견자가 등록된 컴포넌트들에 pinging을 하고 일정 수준 이상 응답이 없는 경우 등록에서 제외시키는 방식을 가지고 있다.

Sensing

SOCAM과 Gaia는 context provider에 의해서 센싱이 이루어지고 CoBrA는 context acquisition module을 통해서 센싱을 한다. CoolAgent RS는 RFID reader와 Web presence manager를 이용해 센싱을 하고 SALSA는 perception context module를 이용해 센싱을 한다.

Context Model

Context model은 context data를 활용, 공유, 저장하기 위한 것으로 상황인식 시스템 작동의 필수요소이다. SOCAM과 CoBrA는 pervasive 컴퓨팅 분야를 여러 소분야로 분리한다. home, office와 같이 분리된 각 소 분야에 대한 하위 온톨로지를 정의하여 복잡도를 줄이고 온톨로지는 OWL로 정의된다. CoolAgent RS는 4가지 분류(the persence of a meeting participant, the profile of a meeting participant, the planned meeting context, organizational information)로 이루어지며 12개의 RDF 스키마 파일로 구현되어 있고 이 스키마 파일에는 agent, documents, meetings, organizations, people 등과 같은 온톨로지로 구성되어 있다. Gaia는 (〈context type〉, 〈subject〉, 〈relator〉, 〈object〉)로 이루어진 4-ary predicate 형식의 온톨로지를 사용하여 context modeling을 한다. 4-ary predicate형식은 context의 표현과 추론규칙 형성에 동일하게 적용된

다. 응용 프로그램들이 상황인지를 하기 위해서는, 우선 상황에 대한 모델이 필요하다. 이를 위해 서술적 문법을 기반으로 한다. 서술적 문법일 기반으로 한 상황 모델은 여러 종류의 메커니즘을 사용하는 상황에 대한 추론의 기본을 제공해 준다. 상황을 표현하는 예는 표 2와 같다.

표 1 Context Predicate Examples

```
Location ( chris , entering , room 3231)
Temperature ( room 3231 , "=" , 98 F)
Sister( venus , serena)
StockQuote( msft , ")" , $60)
PrinterStatus( srgalw1 printer queue , is , empty)
Time( New York , "<" , 12:00 01/01/01)
```

각 술어의 인자의 값은 상황의 형태에 따라 제약사항을 갖는다. 예를 들어, 만약 상황이 Location이라면 첫 번째 인자는 사람 또는 개체가 되어야 한다. 두 번째 인자는 전치사 또는 동사가 되어야 한다.(e.g. "entering", "leaving" or "in") 그리고 세 번째 인자는 위치가 되어야 한다.

Context Processing

센싱과 context model을 통해 얻어진 raw data는 이미 알고 있는 정보를 활용하여 소비자의 관심에 맞게 잘 가공되어 제공되어야 한다. context aggregator가 관심대상인 관련 context 요소의 조합에 연관되어 있다면, context interpreter는 특정 지식을 포함한 context data 변환에 연관되어 있다. context aggregator와 interpreter와 같은 context data abstraction의 형태는 상위 레벨에서의 응용개발을 쉽게 한다. SOCAM은 context reasoning engine을 활용하여 지식베이스 추론, 연역적 context를 유추하는 타스크, context conflict 해결, context 지식베이스의 일관성 유지에 관련된 추론작업을 수행한다. reasoning engine이 활용하는 추론 규칙들이 명시되고, Jena2 [Jena, 2005] 시맨틱 웹 도구를 활용하여 interpreter를 실현하였다. CoBrA 는 context data 처리에 inference engine을 사용한다. inference engine은 context 정보 합성을 위한 Context Reasoning Module을 포함하고 있고, Context Knowledge Base기반 추론과 외부에서 수집된 context 정보로부터 유추되는 추가적 지식 추출을 수행한다. Gaia의 context 처리는 Context Service Module에 내재되어 있다. Context Service Module은 context predicates의 quantification, implication, conjunction, disjunction, negation 과 같은 first order logic operation 수행을 통해 상위수준의 context object 생성을 가능하게 한다.

표 2 Agent 기반의 Context Middleware System

	Architecture	Sensing	Context Model	Resorce discovery
CoBra	agent based	context acquisition module	ontologies (OWL)	n.a.
CoolAg ent RS	agent based	RFID reader, Web presence manager	ontologies (RDF schema)	notification proxy agent(NAP)
Gaia	MVC (extended)	context provider	4-ary predicate (DAML + OIL)	discovery service
SALSA	agent based (scenario based)	context perception	script languages	discovery service
SOCAM	distributed with cen -tralized server	context provider	ontologies (OWL)	service locating service

상황인식 에이전트 시스템을 활용한 대표적인 분야로 특히 헬스-케어 도메인을 들 수 있다. Adapt는 u-Health Care분야의 분산된 애플리케이션 통합을 위한 에이전트 기반의 하부구조인 Agent.Hospital프로젝트의 한 부분으로서 에이전트개발과 실험을 위한 시뮬레이터 연구에 중점을 두었다[23].

u-Health Care 영역은 동적으로 변화하는 상황에 적합한 의료 서비스 (e.g. 환자의 수용계획, 제공되는 의료 서비스의 형식과 정량적인 선택, 긴급구조 시나리오)를 제시할 수 있는 지능적인 서비스들을 필요로 한다. Adapt는 기존의 정보시스템을 통합하는 지능적인 서비스 제공을 위하여 에이전트 기반의 시뮬레이터 기술을 사용하였고 HIS(Hospital Information System)가 구현되지 않은 상태에서 모델링 된 시나리오에 따른 분석 결과를 얻을 수 있다. 그러나 시뮬레이터의 설계 능력에 한정된 모델링만을 수용하고 시뮬레이터가 구현한 에이전트는 특정 플랫폼에 종속적인 면이 있다.

이와 같은 시스템에 에이전트를 설계할 수 있는 정형화된 언어를 도입한다면 앞서 소개된 단점을 극복할 수 있으며 프로그램 없이 에이전트 시스템을 구성할 추상화된 툴 역시 설계 할 수 있을 것이다. SeSAm(ShEll for Simulated Agent systeMs) [24]은 [그림 7]에서 볼 수 있듯이 시각적인 에이전트 모델링과 모델링된 시뮬레이션들을 구동하며 결과의 분석을 지원하는 도구이다. SeSAm은 모의 실험될 에이전트의 특성을 정의하고 UML 활동다이어그램을 이용하여 에이전트의 Behavior를 기술하는 기능을 지원하여 프로그래밍 없이 에이전트를 생성하여 에이전트 기

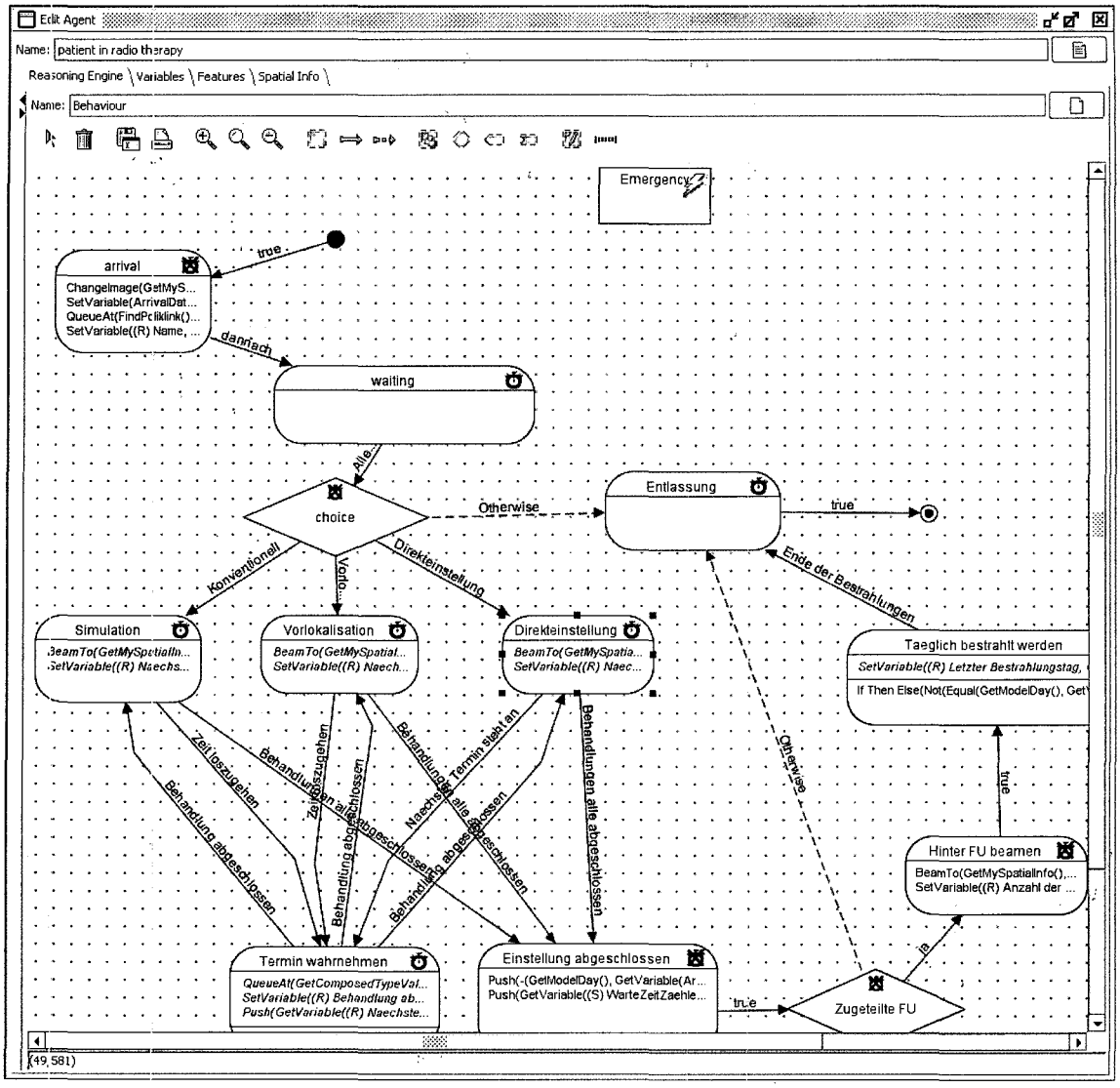

그림 8 SeSAm GUI

반의 애플리케이션을 모의로 제작 하여 준다. 또한, 외부 에이전트와 상호작용을 위하여 FIPA-Communication Primitives와 온톨로지 사용을 지원하며 제작된 에이전트는 멀티에이전트 시스템 환경에서 구동된 결과를 SeSAm에게 전달하여 에이전트 활동의 분석을 가능하게 한다.

에이전트 기반의 상황인식 시스템은 SeSAm에서와 같이 UML과 유사한 범용적인 모델링 언어를 이용하여 에이전트 제작, 분석을 위한 도구의 필요성이 제시된다. 도구 활용을 통하여 Agent-Oriented Software Engineering(AOSE) 방법론을 적용함으로써 유비쿼터스 컴퓨팅에 적합한 시스템의 견고성을 꾀할 수 있다. AOSE 방법론이 적용된 상황 인식 미들웨어를 도입하여 에이전트간 상호운용성을 보장하는 일반적인 방법론을 수용 할 수 있게 된다면 유비쿼터스 컴퓨팅 환경에 구성되는 상황인식 에이전트 시스템의 효율적인 설계와 재사용성이 높아지리라고 본다.

6. 결 론

본고에서는 상황인식 시스템의 디자인 원칙을 조사하고 상황인식 에이전트 시스템의 사례들을 앞서 조사한 디자인 원칙의 견해에서 비교해 보았다. 에이전트 기반 미들웨어 아키텍처의 구체적인 사례로써, 분산형이고 이질적이며 동적요소가 강조되는 유비쿼터스 환경의 상황인식 애플리케에션에 모바일 에이전트 기반 미들웨어가 제공하는 유효한 기능들도 살펴보았다.

상황인식 컴퓨팅에서의 컨텍스트는 사용자와 처리 기기들이 내재되어 있는 물리적이고도 사회적인 상황을 참조하여 추출해 내는 정보이다. 사람, 장소, 시간, 이벤트에 적절한 서비스를 제공하기 위하여 이러한 정보를 수집, 활용하는 것이 상황인식 컴퓨팅의 한 목적이라면 이는 방이나 빌딩들의 공간을 "지능화"하는 단

계를 거쳐 가능하게 된다. 상호작용(interaction)을 통하여 이루어지는 '컴퓨테이션'은 단순히 가상환경의 개인적인 내용만을 다루는 것이 아니라 주변 상황에 영향을 받는 환경적이면서도 집단적인 경험을 다루는 것으로의 변화를 의미한다. 서비스 제공을 목적으로 하는 상황인식 컴퓨팅이 기존의 인터페이스와 상호작용의 기본 개념을 전체적으로 재정의 하는데서 출발하는 반면 현재 대부분의 상황인식 시스템 사례들은 위치인식에 치중된 경향이 있다. 저층과 상위층을 이어주는 미들웨어에서의 code mobility등의 기능 강화로 인해 좀 더 많은 서비스 단의 상황 인지적 시스템의 연구가 활성화되리라고 본다.

이러한 서비스 기반의 에이전트 시스템을 설계하기 위해서 UML(Unified Modeling Language)과 유사한 정형화된 언어를 기반으로 에이전트를 설계하는 AOSE적인 개발 방법론도 고려할 만하다. 정의된 언어를 이용하여 설계된 에이전트는 다중 에이전트 시스템 플랫폼으로 전환되어 구현될 수 있고, 이러한 에이전트 기반 소프트웨어 공학 방법론은 기존의 소프트웨어 공학 방법론에 비하여 더욱 추상적인, 즉 일반적인 설계와 구현 방법론으로 높은 생산성을 이끌어 낼 수 있을 것으로 보여 이 분야의 활성화된 연구도 기대해 본다.

참고문헌

[1] Weiser, M. (1991). The computer for the twenty-first century. Scientific American.

[2] Dey, A. K. and Abowd, G. D. (2000b). Towards a better understanding of context and context-awareness. In Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, New York. ACM Press.

[3] Prekop, P. and Burnett, M. (2003). Activities, context and ubiquitous computing. Special Issue on Ubiquitous Compuitng Computer Communications, 26(11).

[4] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., and Altmann, J. (2002). Context-awareness on mobile devices . the hydrogen approach. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, PP.292-302.

[5] Dey, A. K. and Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting rapid prototyping of context-aware applications. Human-Computer Interactions (HCI) Journal, 16(2-4):7.166.

[6] Ferber, J. (1999) Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. New York: Addison-Wesley.

[7] FIPA, The Foundation for Intelligent Physical Agents, http://www.fipa.org/

[8] W3C (2004a). Composite Capability/Preference Profiles (CC/PP). http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.

[9] Chen, H., Finin, T., and Joshi, A. (2004). An ontology for context-aware pervasive computing environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 18(3):197.207.

[10] Winograd, T. (2001). Architectures for context. Human Computer Interaction (HCI) Journal, 16(2):401.419.

[11] Strang, T. and Linnhoff-Popien, C. (2004). A context modeling survey. In First International Workshop on Advanced Context Modelling, Reasoning And Management, UbiComp 2004.

[12] JADE, Java Agent Development Framework, http://jade.tilab.com/

[13] E. Kovacs, K. Rohrle, M. Reich, "Integrating Mobile Agents into the Mobile Middleware," Proc. *Mobile Agents Int. Workshop* (MA'98), Berlin, Germany, 1998.

[14] P. Bellavista, A. Corradi, C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," *IEEE Computer*, Vol. 34, No. 3, Mar. 2001.

[15] H. Chen, S. Tolia, "Steps Towards Creating a Context-Aware Software Agent System," HP Technical Report HPL-2001-231, Sep. 2001

[16] P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In H.-W. Gellerson, editor, Handheld and ubiqitous computing, number 1707 in Lecture Notes in Computer Science, pages 304{7. Springer, September 1999.

[17] Gu, T., Pung, H. K., and Zhang, D. Q. (2004a). A middleware for building context-aware mobile services. In Proceedings of IEEE Vehicular Technology Conference (VTC), Milan, Italy.
[18] Chen, H. (2004). An Intelligent Broker Architecture for Pervasive Context-Aware Systems. PhD thesis, University of Maryland, Baltimore County.
[19] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, Klara Nahrstedt : Gaia: A Middleware Infrastructure to Enable Active Spaces : 2-3, 7/1/2002
[20] Marcela Rodriuez and Jesus Favela, "A Framework for Supporting Autonomous Agents in Ubiquitous Computing Environments," CICESE, Ensenada, Mexico, 2002.
[21] Salber, D., Dey, A. K., and Abowd, G. D. (1999). The Context Toolkit: Aiding the development of context-aware applications. In Proceedings of the ACM CHI, Pittsburgh, PA.
[22] Jeremy J. Carroll,Dave Reynolds "Jena: Implementing the Semantic Web" Recommendations HP Labs, Bristol UK 2005.
[23] Stefan Kirn, Christian Heine, Rainer Herrler, Karl-Heinz Krempels. Agent.Hospital - agent-based open framework for clinical applications, wetice, p.36, Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.
[24] Franziska Klügl, Rainer Herrler, and Christoph Oechslein, "From Simulated to Real Environments: How to Use SeSAm for Software Development," Universität Würzburg Am Hubland, 97074 Würzburg.

김 지 환

2000~현재 가톨릭대학교 컴퓨터공학과 학부과정
관심분야 : 지능형 에이전트, 감성공학, 시맨틱 웹서비스
E-mail : bangga486@catholic.ac.kr

양 정 진

가톨릭대학교 컴퓨터·정보공학부
1985 이화여자대학교 전자계산학과(학사)
1992 University of Connecticut(공학석사)
1998 University of Connecticut(공학박사)
1999 University of Connecticut Post Doc.
1999~2000 University of Hartford 조교수
2001~현재 가톨릭대학교 컴퓨터·정보공학부 부교수
관심분야 : 지능형 (다중)에이전트 시스템, 유비쿼터스 컴퓨팅, 시맨틱웹서비스
E-mail : jungjin@catholic.ac.kr