# 유비쿼터스 컴퓨팅에서 상황인식 에이전트를 위한 미들웨어

배 인 한*

◈ 목 차 ◈



## 1. 서 론

유비쿼터스 환경에서는 언제 어디서든지 컴퓨터에 접근이 가능하기 때문에, 사람의 위치를 파악하고 현 상황에 맞는 적절한 서비스를 제공해 주게 된다. 이러한 적절한 서비스를 제공하기 위하여 스스로 판단하여 행동할 수 있는 에이전트를 통한 커뮤니케이션이 필요하다. 에이전트는 사용자나 다른 에이전트의 직접적인 지시나 간섭 없이 능동적으로 작업수행을 진행한다. 그리고 사용자가 제공받고자 하는 서비스나 목적달성을 위한 세부절차 등을 맡게 된다. 에이전트가 작업을 수행하면서 필요에 따라 에이전트 간의 메시지 교환을 통하여 다른 에이전트의 도움을 받는다[1]. 유비쿼터스 컴퓨팅 환경은 물리적 공간을 스마트한 대화형 환경으로 변환하기 위하여 함께 작업하는 다수의 자치적인 에이전트들로 구성된다. 어떤 에이전트가 유비쿼터스 환경에서 효율적으로 작동하기 위하여, 2가지 태스크를 수행해야 한다. 에이전트들은 환경의 현재 상황에 대한 지각과 추론을 해야 하고, 에이전트들은 다른 에이전트들과 원활하게 상호작용해야 한다.

본 논문에서, 우리는 유비쿼터스 컴퓨팅 환경에서 에이전트의 상기 2가지 요구사항을 만족시키는 미들웨어를 소개한다. 만일 자동화된 에이전트들이 작동중에 상황에 대해 지각하고 추론할 수 있다면 그 자동화된 에이전트 역시 상황을 고려하여 적절한 행위를 수행할 수 있다. 유비쿼터스 컴퓨팅 환경은 다양한 다른 상황을 지각할 수 있는 다수의 센서들에 의해 특징 지워진다. 상황의 종류는 물리적 상황(위치, 시간), 환경적 상황(날씨, 조명과 소리 정도), 정보 상황(주식 시세, 스포츠 스코어), 개인적 상황(건강, 감정, 스케줄, 활력), 사회적 상황(그룹 활동, 사회적 관계, 방내의 다른 사람), 응용 상황(수신된 이메일, 방문한 웹사이트) 그리고 시스템 상황(망 트래픽, 프린터 상태)을 포함한다. 에이전트들이 행동하는 방법을 환경에 적응시키기 위하여 상황을 획득하고 추론해야 한다. 우리는 상황인식을 위하여 미들웨어가 제공해야하는 유비쿼터스 컴퓨팅 환경을 논의한다. 상황인식을 위한 미들웨어는 상황을 처리하는데 포함되는 대부분의 작업을 위한 지원을 제공할 것이다. 상황인식 에이전트들은 그러한 미들웨어로 매우 쉽게 개발될 수 있을 것이다. 상황인식을 위한 미들웨어는 상황을 처리하기 위한 에이전트의 배치에서 다른 메커니즘을 위치시킬 수도 있다. 그러한 메커니즘들은 다른 종류의 논리(일차 논리, 시간 논리, 퍼지 논리)로 작성된 규칙과 같은 추론 메커니즘뿐만 아니라 베이지안 네트워크, 신경망 또는 강화 학습과 같은 학습 메커니즘도 포함한다. 상황인식 에이전트 개발자들은 다른 센서들로부터 얻은 상황 정보의 복잡한 항목 또는 상황 추론을 위하여 추론 메커니즘 또는 학습 메커니즘에 대해 걱정하지 않아도 된다. 에이전트들은 그것들의 추

* 대구가톨릭대학교 컴퓨터정보통신공학부

론하는 요구조건을 가장 잘 만족시키는 적합한 논리를 선택할 수 있다. 유비쿼터스 컴퓨팅 환경에서 미들웨어의 다른 중요한 요구사항은 자치적인 이형의 에이전트들이 다른 에이전트와 미끄럽게 상호작용하는 것을 허용하는 것이다. TCP/IP, CORBA, Jini, SOAP 등과 같은 다수의 프로토콜들과 미들웨어들은 분산 에이전트들이 서로 통신할 수 있도록 개발되었지만, 그것들은 에이전트들 간의 신태틱 그리고 시맨틱 상호운용성 문제를 처리하지 못하였다. 그것들은 서로 상호작용할 때 에이전트들이 사용할 수 있는 공통 용어와 개념들의 공유 집합을 제공하지 않았다. 이 문제는 다른 에이전트들이 현재 상황을 다르게 이해할 수 있기 때문에 상황 정보 분야에서 특히 심각하다. 그것들은 상황을 설명하기 위하여 다른 용어를 사용할 수 있고, 만일 그것들이 같은 용어를 사용할지라도, 그것들은 그러한 용어들에 다른 의미를 부여할 수도 있다. 상황인식을 위한 미들웨어는 에이전트들이 상황 정보를 교환할 때 다른 에이전트들 간의 시맨틱 차이가 없도록 보장하여 이 문제를 처리해야 한다.

본 논문의 구성은 다음과 같다. 2장에서는 유비쿼터스 컴퓨팅에서 필수적으로 요구되는 상황인식과 관련된 기술과 기존의 상황인식 미들웨어에 대하여 살펴보고, 3장에서는 상황 정보에 대한 높은 수준의 추상적 개념을 제공하는 상황 모델을, 4장에서는 에이전트들에게 다양한 상황 정보를 획득하고 추론을 하는 방법들을 제공하는 상황인식 인프라구조를, 그리고 5장에서는 상황인식 에이전트를 위한 미들웨어의 구성요소들에 대해서 설명한다. 그리고 마지막으로 6장에서 결론 및 향후 연구내용에 대하여 기술한다.

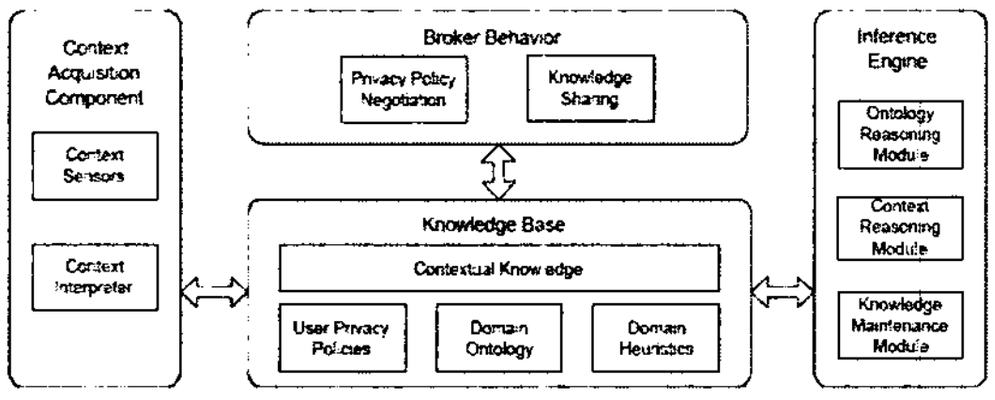
## 2. 관련연구

과거 몇 년 동안에 상황인식 컴퓨팅 분야에 많은 연구가 수행되었다. 상황인식 컴퓨팅 정의하는 것, 상황인식 응용 구축을 위하여 어떤 지원이 요구되는지 식별하는 것, 그리고 상황인식 응용의 빠른 원형을 만드는 툴킷을 개발하는 것에 대한 기본적인 연구가 Anind Dey 등에 의해 수행되었다[1]. Context Toolkit은 상황 정보를 사용하기 위하여 응용에 대한 출발점을 제공하지만, 상황에 관하여 어떻게 추론할지에 대한 많은 도움을 제공하지 못했다. 그것은 상황에 대한 규칙을 작성하는 것, 고급 상황을 추론하는 것 또는 광범위한 가능 상황을 구조화된 형식으로 구성하기 위한 일반적인 메커니즘을 역시 제공하지 못했다. [2]에서, Jason Hong 등은 툴킷과 인프라구조를 구별하였다. Hong에 따르면 인프라구조는 다른 시스템들을 위한 기초를 제공하는 잘 설치된 편재하는 신뢰성 있는 기술들의 집합이다. 상황인식을 위한 미들웨어는 Hong의 인프라구조 개념위에 구축되고, 상황인식 응용을 쉽게 개발하도록 기초를 제공한다. Bouquet 등[3]은 자치적이고 이질적인 분산 응용에서 상황 문제를 다루었다. 여기서 각 개체는 그것의 관점에 의존하는 상황에 속하는 자신의 표기법을 갖는다. 다른 개체들과 상호작용하기 위하여, 어떤 개체는 그것의 관점과 다른 개체의 관점 간의 관계를 알아야 한다. 미들웨어는 좀 더 일반적인 방식으로 이 상호운용성을 성취하기 위하여 온톨로지를 사용한다. Paul Castro와 그의 동료들[4]은 베이지안 네트워크를 사용하여 센서 데이터로부터 유용한 상황 정보를 추출하고 추론하는 "fusion services" 개발에 대해 연구하였다. 미들웨어는 그러한 학습 방법들이 제공되는 좀 더 일반적인 프레임워크를 제공해야한다. Terry Winograd는 상황에 대한 다른 구조들을 비교하고 집중화된 Event Heap을 사용하는 구조를 제안하였다[5]. [6]에서, Brumitt 등은 상황인식 환경에서 멀티 모달 상호작용을 갖는 경험과 어떤 환경이 다른 상황에 어떻게 자동적으로 응답할 수 있는지를 설명하였다. 미들웨어는 개발자들에게 어떤 환경이 다른 상황에 어떻게 자동적으로 응답할 수 있는지를 명시하는 쉬운 방법을 제공해야한다.

재구성 가능한 상황 센시티브 미들웨어는 상황 센시티브 응용들에게 런타임 상황 데이터 획득, 감시 그리고 탐지를 위한 적응적 객체 컨테이너를 제공한다. 응용들은 상황인식 IDL(Interface Definition Language)을 사용하여 행위를 명세할 수 있다. 미들웨어는 다른 추론 메커니즘과 학습 메커니즘을 사용하여 상황인식 응용들의 행위를 명시하는 좀 더 일반적인 방법을 제공해야한다.

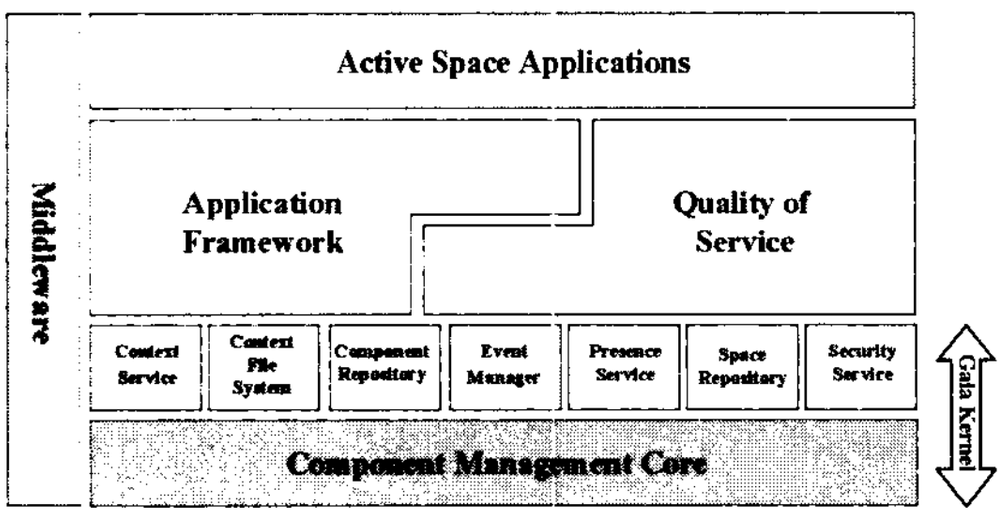
CoBra(Context Broker Architecture)[7]는 에이전트 기반 지능형 공간에서 상황인식을 제공하기 위한 구조

를 가지고 있다. CoBra 시스템에서 지능형 공간은 물리적 공간으로 지능형 시스템이 공간에 스며들어 사용자에게 서비스를 제공한다. 지능형 상황 브로커는 에이전트 커뮤니티 사이에서 상황 모델을 공유, 관리한다. 각 에이전트는 사용자가 휴대하는 모바일 기기에서 동작하는 응용일 수 있고, 각 상태 정보를 웹상에서 제공하는 웹 서비스일 수도 있다. 브로커는 상황 지식 베이스, 상황 추론 엔진, 상황 획득 모듈, 보안 관리 모듈로 구성되어 있다.



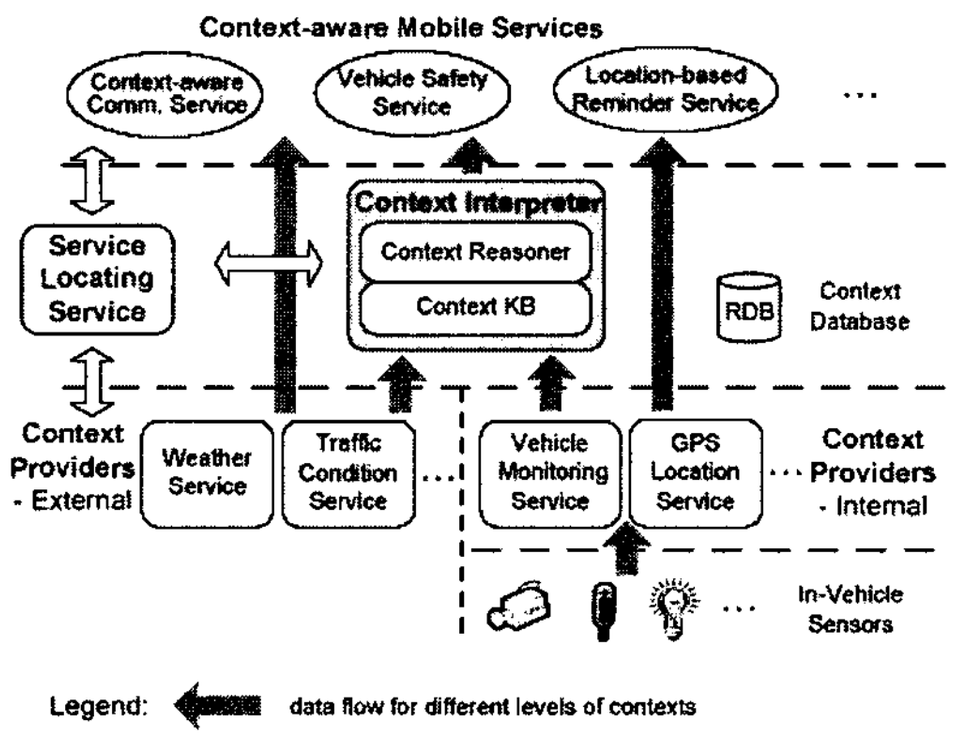

(그림 1) CoBra의 개념적 설계

Gaia[8]는 상황인식 서비스 구조로 응용이 다양한 상황정보를 얻고 추론할 수 있게 해주며, 상황 처리를 위해 논리 추론과 기계 학습 방법을 폭넓게 활용한다. 서로 다른 유비쿼터스 컴퓨팅 환경뿐만 아니라 이종 에이전트간의 시맨틱한 상호운용성을 보장하기 위하여 GPDL(Generalized Policy Definition Language)과 사용자 인식, 시간, 장소 등에 따라 적절한 권한 부여를 위해 GRBAC(Generalized Role-Based Access Control) 모델을 제안하였다.




(그림 2) Gaia의 구조

SOCAM (Service-Oriented Context-Aware Middleware) [9]는 상황인식 모바일 서비스의 구축과 빠른 원형 생성을 제공한다. 상황인식 모바일 서비스는 아키텍처의 최상위에 위치하여 각 서비스는 서로 다른 상황정보를 활용하고 이에 따라 적응력 있게 에이전트의 행위를 수행한다.




(그림 3) SOCAM 구조의 개요

SALSA 구조[10]는 명명, 등록, 인증, 에이전트의 지역화, 간단한 API가 자율적인 에이전트 개발을 가능하게 하는 서비스의 인프라구조를 제공한다. SALSA는 모바일 사용자를 위한 단절 모드 연산을 지원하고 사용자와 기기 사이의 상호운용성을 위해 현재 존재하거나 새로운 서비스 간의 상호작용 메커니즘을 가진다. 에이전트들 간의 상황인식을 촉진하기 위하여 다른 방법들이 제안되었다.

Anind Dey et al [1]은 센서 기반 상황인식 응용의 개발과 실행을 위한 프레임워크를 제공하고 다수의 재사용 가능한 컴포넌트들을 제공하는 Toolkit 방법을 제안하였다. 그 Toolkit은 어떤 종류의 상황인식 응용의 빠른 원형 생성을 지원한다.

미들웨어는 상황인식 시스템을 생성하고 관리하는 일을 아주 간단히 한다. 미들웨어는 공통적인 연산에 대한 동일한 추상화와 신뢰성 있는 서비스를 제공한다. 그것은 유비쿼터스 환경에서 새로운 센서 에이전트와 상황인식 에이전트 개발을 쉽게 만든다. 미들웨어는 하드웨어, 운영체제 그리고 프로그래밍 언어에 독립적이다. 결국, 미들웨어는 다수의 상황인식 에이전트들 간의 상호작용에 기반 한 복잡한 시스템을 구성할 수 있다. CORBA와 Jini와 같은 전통적인 미들웨

어는 서로 통신하기 위하여 다른 객체들 또는 에이전트들을 위하여 기본 메커니즘을 제공하지만, 그것들은 에이전트에 상황인식 기능을 제공하는 방법에 단점이 있다. 상황인식은 상황 정보의 획득, 상황에 대한 추론 그리고 현재 상황에 기초하여 에이전트의 행위 수정을 포함한다. 상황인식에 대한 미들웨어는 그러한 태스크에 대한 지원을 제공한다. 미들웨어는 모든 에이전트들이 상황 처리에 사용할 수 있는 상황의 공통 모델을 정의한다. 미들웨어는 유비쿼터스 환경에서 다른 에이전트들이 상황 정보에 대한 공통적인 시맨틱 이해를 가지는 것을 보장한다. 상황인식을 위한 미들웨어는 분산 에이전트들이 서로 검색하고 통신이 가능하도록 CORBA를 사용한다. 미들웨어는 상황인식이 가능하도록 여분의 기능을 제공한다. 그것은 에이전트들이 상황 정보를 획득하고 추론하도록 다양한 서비스와 라이브러리를 제공한다.

(표 1) 에이전트 기반 상황인식 미들웨어 시스템들

| 특징 / 미들웨어 | 구조 | 센싱 | 상황 모델 | 자원 발견 |
|---|---|---|---|---|
| Cobra | 에이전트 기반 | 상황 획득 모듈 | 온톨로지 (OWL) | 지원 안됨 |
| Gaia | MVC(Model View Controller) | 상황 제공자 | 4-차 술어 (DAML+OIL) | 발견 서비스 |
| SALSA | 에이전트 기반 | 상황 지각 | 스크립트 언어 | 발견 서비스 |
| SOCAM | 집중형 서버를 갖는 분산 방식 | 상황 제공자 | 온톨로지 (OWL) | 서비스 검색 서비스 |

# 3. 상황모델

상황의 역할은 유비쿼터스 컴퓨팅 분야에서 최근에 아주 중요해 지고 있다. "상황(context)"은 사용자와 유비쿼터스 컴퓨팅 환경 사이의 상호작용과 관련이 있는 어떤 사용자를 둘러싼 환경, 객체들, 또는 조건들에 대한 어떤 정보이다[1]. 유비쿼터스 컴퓨팅 환경에서 상황인식 응용을 만들려는 다수의 연구가 수행되고 있다. 따라서 상황인식 응용들은 다른 상황에 적응할 수 있고, 사용자의 요구에 대해 더 큰 적응력을 갖는다[1, 2, 6, 11, 12]. 인간은 다른 상황에서 다르게 행동한다. 인간들은 상황이 무엇인지 지각할 수 있고, 행위를 현재 상황에 적응시킨다. 인간이 스스로 적응하는 방법은 경험을 통하여 배운 규칙에 기초한다. 인간은 과거 경험과 현재 상황에 맞는 사회적 정치적으로 정당한 행위를 수행할 수 있다. 상황 모델은 상황 정보를 활용, 공유, 저장하기 위한 것으로 상황인식 시스템 작동의 필수적인 요소이다.

## 3.1 상황술어

응용이 상황인식을 하기 위하여, 먼저 상황에 대한 모델이 필요하다. 술어에 기초한 상황 모델을 개발한다. 다른 상황 술어의 속성과 구조를 기술하기 위하여 온톨로지를 사용한다. 이런 상황 모델은 다양한 메커니즘을 사용하여 상황에 대한 추론을 위한 기초를 제공한다. 어떤 상황은 어떤 술어로 표현한다. 술어명이 장소, 온도 또는 시간과 같은 기술되어질 상황의 종류인 어떤 규약에 따른다. 이 규약은 다른 종류의 상황들을 위해 간단하고, 같은 형식의 표현을 허용한다. 이 외에, 그것은 온톨로지에 다른 상황을 쉽게 기술할 수 있고, 술어의 인자로 "="와 "<"와 같은 관계 연산자를 사용할 수 있다.

상황 술어의 예는 표 2와 같다.

(표 2) 상황 술어의 예

- 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 인자 값은 상황의 형태에 따라 제약사항을 갖는다. 예를 들어, 만약 제약 사항이 장소(location)이라면, 첫 번째 인자는 사람 또는 객체가 되어야 한다. 두 번째 인자는 "들어가다(entering)", "떠나다(leaving)" 또는 "~안에(in)"와 같은 전치사 또는 동사가 되어야 한다. 그리고 세 번째 인자는 장소가 되어

야 한다. 그 술어가 올바르다는 것을 확인하기 위하여 형 검사를 수행한다.

## 3.2 상황 술어 기술을 위한 온톨로지

다른 상황 술어의 구조는 온톨로지로 명세 되었다 [13]. 온톨로지의 특성을 잘 표현하고 있어 가장 널리 받아들여지고 있는 그루버(T. Gruber)의 온톨로지 정의는 다음과 같다. "온톨로지란 관심 영역 내 공유된 개념화에 대한 형식적이고 명시적인 명세화이다(An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest)" 온톨로지는 데이터베이스의 일종이라 할 수 있는데, 이 데이터베이스에는 보통의 관계형 데이터베이스의 경우와는 달리 개념들 간 위계 구조와 기타 다른 관계 및 제약이 표현되어 있다. 각 상황 종류는 온톨로지에서 클래스에 대응한다. 온톨로지는 다양한 상황 종류뿐만 아니라 그 술어들이 가져야 하는 인자들을 정의한다. 온톨로지는 시맨틱 웹의 사실상 언어인 DAML+OIL [14]로 작성된다.

예를 들어, 다수의 상황 술어들은 인자를 가지기 위하여 SVO(Subject Verb Object) 형식으로 정의된다. 즉, 그러한 술어들의 구조는 ContextType(<Subject>, <Verb>,<Object>)이다. 실례로, 온톨로지의 Location 술어는 사람들 또는 사물들의 집합이 속하는 주체, "inside" 또는 "entering"과 같은 전치사 그리고 방 또는 빌딩이 될 수 있는 위치를 가져야 한다.

온톨로지는 상황 술어의 유효성을 검사하는데 사용된다. 그것은 역시 다른 상황 술어 작성을 더 쉽게 만든다. 따라서 그 술어의 구조가 무엇인지 그리고 다른 인자의 어떤 종류 값이 발생할 수 있는지를 안다. 그것은 역시 다른 유비쿼터스 컴퓨팅 환경이 상호 작용하는 것을 허용한다. 따라서 그것은 그러한 환경의 온톨로지에서 사용되는 용어들 간의 변환을 정의하는 것이 가능하다.

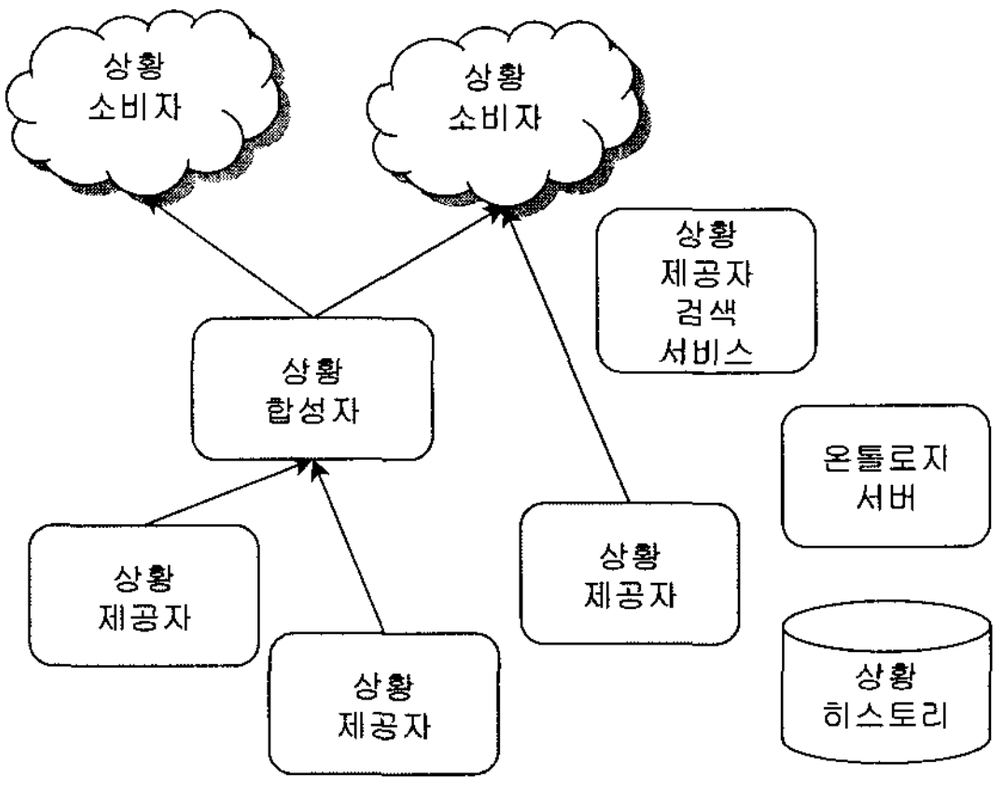
상황을 위한 논리적 모델은 아주 강력하다. 그것은 프로그래밍 언어, 운영체제 또는 미들웨어에 독립적인 일반적인 방법으로 어떤 시스템의 상황을 기술하는 것을 허용한다. 상황 술어의 구조와 신맨틱스는 온톨로지로 기술되므로 그것은 그 시스템내의 다른 구성요소들이 다른 상황들의 시맨틱의 공통적인 이해를 가지는 것을 허용한다.

상황에 관한 술어 모델은 다른 추론 메커니즘을 허용할 만큼 충분히 일반적이다. 예를 들어, 일차 논리 또는 시간 논리와 같은 다른 논리들을 사용하여 응용 행위를 기술하는 그러한 상황 술어를 사용하여 규칙 작성하는 것이 가능하다. 베이지안 네트워크, 신경망 또는 다른 방법들을 사용하여 그러한 술어들에 기반한 다른 종류 추론을 수행하는 것이 가능하다.

# 4. 상황인식 인프라구조

상황인식 인프라구조는 다양한 상황 정보를 획득하고 그것에 관한 추론을 하기 위하여 에이전트들에게 다양한 방법들을 제공한다. 상황인식을 위한 인프라구조의 다이어그램은 그림 5와 같다[15].

- 상황 제공자(Context Providers): 상황 제공자는 센서들 또는 상황 정보의 다른 데이터 소스들이다. 상황 제공자들은 다른 에이전트들이 상황 정보를 위해 상황 제공자들을 질의하는 것을 허용한다. 몇몇 상황 제공자들은 상황 이벤트 전송을 관리하기 위하여 이벤트 채널을 역시 가지고 있다. 즉, 다른 에이전트들은 어떤 제공자에게 질의하거나 상황 정보를 얻기 위하여 이벤트 채널을 도청한다.




(그림 4) 상황인식 인프라구조

- 상황 합성자(Context Synthesizers): 상황 합성자는 다양한 상황 제공자들로부터 감지된 상황을 얻고, 그러한 간단한 감지된 상황으로부터 고급 또는 추상 상황을 추론하고, 그리고 다른 에이전트들에게 그러한 추론된 상황을 제공한다. 예를 들어, 방의 사람 수와 실행하는 응용에 기초하여 방에서 진행중인 활동을 추론하는 어떤 상황 합성자가 있을 수 있다.
- 상황 소비자(Context Consumers): 상황 소비자 또는 상황인식 응용은 상황 제공자 또는 상황 합성자로부터 다른 종류의 상황들을 얻을 수 있는 에이전트들이다. 그것들은 현재 상황에 대해 추론하고 그 현재 상황에 따라 그것들이 행동하는 방법을 적응시킨다.
- 상황 제공자 검색 서비스(Context Provider Lookup Service): 상황 제공자는 상황 제공자 검색 서비스를 제공한다는 것을 광고한다. 이 서비스는 에이전트들이 적절한 상황 제공자를 찾는 것을 허용한다. 여기에는 하나의 유비쿼터스 컴퓨팅 환경에 하나의 그러한 서비스가 있다.
- 상황 히스토리 서비스(Context History Service): 과거 상황들은 데이터베이스에 기록되었다. 상황 히스토리 서비스는 다른 에이전트들이 과거 상황을 질의하는 것을 허용한다. 여기에는 하나의 유비쿼터스 컴퓨팅 환경에 하나의 그러한 서비스가 있다.
- 온톨로지 서버(Ontology Server): 온톨로지 서버는 다른 종류의 상황 정보를 기술하는 온톨로지를 관리한다. 여기에는 유비쿼터스 컴퓨팅 환경 마다 하나의 온톨로지 서버가 있다.

## 5. 상황인식 미들웨어

유비쿼터스 컴퓨팅 환경에서 상황인식을 위한 미들웨어를 위한 다수의 요구사항들은 다음과 같다.

(1) 다른 센서들로부터의 상황 정보의 수집과 다른 에이전트로의 상황 정보의 전달을 지원
(2) 저수준 감지된 상황으로부터 고수준 상황을 추론 하는 것을 지원
(3) 에이전트들이 여러 가지 추론과 학습 메커니즘을 사용 가능하게 하는 것.
(4) 에이전트들이 다른 상황에서 다른 행동을 명시하는 것을 허용하는 기능
(5) 다른 에이전트들 간의 구문적 의미적 상호운용성을 가능하게 하는 것

상황인식 미들웨어의 핵심 특징은 에이전트들에게 상황에 대한 적절한 추론을 지원하기 위한 다양한 종류의 추론 메커니즘 그리고/또는 학습 메커니즘을 제공하는 것이다. 그러한 추론 또는 학습 메커니즘을 사용하여 에이전트들은 현재 상황에 대한 다양한 속성을 추론하고, 상황에 대한 논리 질의에 응답하고 또는 에이전트들이 다른 상황에서 행동하는 방법에 적응할 수 있다. 에이전트들은 일차 논리, 시간 논리, 기술 논리, 고차 논리, 퍼지 논리 등과 같은 다른 종류의 논리로 작성된 논리를 사용하여 상황에 대한 추론을 할 수 있다. 다양한 사건이 발생하는 시간 순서와 관련된 에이전트는 규칙을 표현하기 위하여 몇몇 형태의 시간 논리 사용이 필요할 것이다. 존재 또는 보편적 기호를 사용하여 일반 조건을 표현하는 것이 필요한 에이전트들은 몇몇 형태의 일차 논리가 필요할 것이다. 표현력을 더 요구하는 에이전트는 고차 논리를 요구할 수 있다. 술어상의 계층구조 명세를 처리하는 에이전트들은 술어 논리가 필요할 수도 있다. 불확실성을 처리하기 위한 에이전트들은 몇몇 형태의 퍼지 논리를 요구할 수도 있다. 상황에 대한 추론을 위해 몇몇 형태의 논리로 작성된 규칙 대신에, 에이전트들은 상황을 처리하기 위하여 다양한 기계 학습 기법들을 사용할 수도 있다. 사용될 수 있는 학습 기법들은 베이지안 학습, 신경망, 강화 학습 등을 포함한다. 학습되어질 개념의 종류에 따라 다른 학습 메커니즘이 사용될 수 있다. 만일 어떤 에이전트가 대화형 방식으로 다른 상태에서 수행할 적절한 행위를 학습하기를 원한다면, 그것은 강화 학습 또는 신경망을 사용한다. 만일 어떤 에이전트가 다른 이벤트의 조건 확률을 학습하기를 원한다면, 베이지안 학습이 적합하다. 어떤 종류의 논리 또는 학습 메커니즘을 사용할지에 대한 결정은 그 논리의 표현력뿐만 아니라 성능, 용이성 그리고 결정성과 같은 다른 문제에 의존한다. 미들웨어

는 에이전트들이 상황을 이해하고 반응하는데 사용할 수 있는 추론 메커니즘과 학습 메커니즘의 선택을 에이전트들에게 제공한다.

## 5.1 상황 제공자

미들웨어는 위치, 날씨, 주식 가격, 일정표와 스케줄 정보 등과 같은 다양한 종류의 상황을 제공하는 인프라구조에서 다수의 상황 제공자를 가지고 있다. 다른 상황 제공자들은 그것들이 감지한 상황에 대한 추론을 위하여 그리고 질의에 응답하기 위하여 다른 추론 메커니즘과 학습 메커니즘을 사용한다. 예를 들어, 불확실한 상황을 처리하는 상황 제공자는 퍼지 논리를 사용하는 반면에, 변수를 정량화하는 기능을 요구하는 상황 제공자는 일차 논리를 사용할 수 있다. 위치 상황 제공자는 일차 논리를 사용하므로 사람 또는 위치를 정량화할 수 있다. 그것은 방의 모든 사람 또는 빌딩내의 모든 방에 대한 질의에 응답할 수 있다. 날씨 예보 상황 제공자는 다른 상황에 확률을 첨부하기 위하여 어떤 형태의 퍼지 논리를 사용한다. 예를 들어 내일 강수량이 어떤 확률로 발생할 것 수 있다고 말한다. 상황 제공자는 현재 상황을 얻기 위하여 다른 에이전트들을 위해 질의 인터페이스를 제공한다. 그 결과는 "예(yes)" 또는 "아니오(no)"이다. 만일 질의가 변수를 갖는 술어이면, 그 결과는 그 술어를 참으로 만드는 상수를 갖는 변수의 어떤 통합을 포함해야 한다. 반환되는 통합된 상황 술어들은 사용된 논리 종류에 의존하여 시간 또는 확률과 같은 부가적인 속성을 가질 수도 있다.

몇몇 상황 제공자들은 어떤 이벤트 채널로 그것들의 상황에 대한 이벤트를 역시 전송한다. 소비자들은 이 채널을 도청할 수 있다. 예를 들어, 방 기반 위치 서비스는 Bob이 방 3231에 입실하였을 때 "Location (Bob, Entering, Room 3231)"과 같은 이벤트를 전송한다. 엄밀히, 어떤 상황 제공자가 생성되었을 때 어떤 이벤트가 상황 제공자를 위해 어떤 정책에 의해 설정되어진다. 다소의 경우에, 그 제공자는 이벤트를 주기적으로 전송하는 것을 유지한다. 예를 들어, 날씨 서비스는 매 5분마다 온도 최신정보를 전송을 유지한다. 다른 경우에, 그 제공자는 상황에서 변화가 탐지되자마자 이벤트를 전송한다. 모든 상황 제공자는 상황을 획득하고 상황 이벤트를 도청하기 위한 유사한 인터페이스를 지원한다. 따라서 소비자들은 그것들이 질의하는 상황 제공자의 실제 종류에 대해 걱정할 필요가 없다. 이것은 상황인식 응용의 개발에 크게 도움이 된다.

## 5.2 상황 합성자

상황 합성자는 간단한 감지된 상황에 기초하여 고급 상황을 제공하는 에이전트이다. 상황 합성자는 다양한 상황 제공자로부터 소스 상황을 획득하고, 그것들에 다소의 논리를 적용하고, 새로운 종류의 상황을 생성한다. 상황 합성자는 상황 제공자이고 상황 소비자이다. 상황 제공자와 유사하게, 상황 합성자는 다른 에이전트가 현재 상황을 획득하는데 사용할 수 있는 Prolog와 비슷한 질의 인터페이스를 역시 지원한다. 그것들은 어떤 이벤트 채널로 이벤트들을 역시 전송할 수도 있다. 기존 상황으로부터 새로운 상황을 추론하기 위한 2가지 기본적인 방법이 있다. 첫 번째는 고급 상황을 추론하기 위하여 정적 규칙을 사용하는 것이고 다른 것은 기계 학습 기법을 사용하는 것이다.

규칙 기반 합성자는 다른 상황을 추론하기 위하여 몇몇 형태로 작성된 미리 정의된 규칙들을 사용한다. 예를 들어, 방안의 사람의 수와 방에 기초하는 Room Activity Context Provider를 가지고 있다. 그 방에서 실행중인 응용은 그 방에서 어떤 종류의 활동이 일어나고 있는지 추론한다. 그 추론을 수행하기 위하여 일차 논리로 작성된 규칙을 사용한다. 이 Room Activity Context Provider가 사용하는 몇몇 규칙은 표 3과 같다.

(표 3) Room Activity Context Provider의 상황 합성을 위한 규칙의 예

(1) #People( Room 2401, ">=", 3) AND Application( PowerPoint,, Running) => RoomActivity( 2401, Presentation)
(2) #People( Room 2401, ">=", 1) AND Application( MPEG Player, Running) => RoomActivity( 2401, Movie Screening)
(3) #People( Room 2401, ">=", 3) AND NOT ∃ $_{\text{Entertainment-Application}}$ x Application (x, Running) => RoomActivity( 2401, Meeting)

(4) #People( Room 2401, ">=", 3) AND Application( PowerPoint,, Running) => RoomActivity( 2401, Presentation)
(5) #People( Room 2401, ">=", 1) AND Application( MPEG Player, Running) => RoomActivity( 2401, Movie Screening)
(6) #People( Room 2401, ">=", 3) AND NOT ∃ Entertainment-Application x Application (x, Running) => RoomActivity( 2401, Meeting)
(7) #People( Room 2401, "=", 1) AND Application( Visual Studio, Running) => RoomActivity(2401, Individual Development)
(8) #People( Room 2401, "=", 2) AND Application( Visual Studio, Running) => RoomActivity(2401, Extreme Programming)
(9) #People( Room 2401, "=", 0) => RoomActivity( 2401, Idle)

각 규칙들은 그것과 관련된 우선순위를 역시 가지고 있다. 하나 이상의 규칙들이 동시에 참이면, 규칙들의 우선순위를 사용하여 그 방에서 활동이 정확하게 결정되어진다. 만일 2개의 규칙이 동시에 참이고 같은 우선순위를 갖는다면, 그것들 중 하나가 무작위로 선택되어진다.

규칙 기반 합성자들은 사람에 의한 규칙들의 분명한 정의를 요구한다는 단점이 있다. 그것들은 역시 유연하지 못하고 환경 변화를 수용할 수 없다. 고급 상황을 추론하기 위하여 기계 학습 기법의 사용하여 이 문제를 극복할 수 있다. 감지하기 매우 어려운 한 가지 상황은 사용자의 감성이다. 각 사용자가 다르기 때문에 사용자 감성을 예측하기 위한 규칙들을 작성하는 것은 어렵다. 사용자의 감성에 영향을 줄 수 있는 아주 많은 요인들이 있다. 위치, 현재시간, 그와 함께 방에 있는 다른 사람, 바깥 날씨 그리고 그의 주식 투자가 어떻게 되어 가는지와 같은 다소의 가능 요인을 고려한 학습 메커니즘을 사용할 수 있다. Uesr Mood Context Provider는 사용자 감성을 예측하기 위하여 Naive Bayes 알고리즘을 사용한다. 학습자를 훈련시키기 위하여 과거 상황을 사용한다. 훈련 단계 동안에, 그의 감성에 대해 주기적으로 물어본다. 그 사용자가 그의 감성에 들어갈 때 마다 특징 값(예, 위치, 날씨 등)을 발견하여 훈련 예를 구축한다. 일주일 동안 학습자를 훈련한다. 훈련 단계가 끝나면, 학습자는 미래 상황의 값을 나타내는 사용자의 감성을 예측할 수 있다. Naive Bayes의 결과는 확률적이다. 다른 사용자 감성과 상황 처리에서 몇몇 형태의 퍼지 논리 또는 확률적 추론과 관련된 확률을 얻는 것이 가능하다.

## 5.3 상황 소비자(상황인식 응용)

상황 소비자는 다양한 종류의 상황을 소비하고 현재 상황에 따라 그것들의 행위를 적응시키는 에이전트이다. 응용을 상황에 민감하게 만들 수 있는 한 가지 일반적인 방법은 환경의 상황이 바뀌었을 때 수행되어질 행위를 명시하는 것이다. 즉, 환경의 상황이 변경되자마자, 그 응용은 새로운 상황의 요구사항을 만족시키기 위하여 스스로 재구성된다. 예를 들어, 어떤 스마트 룸의 주크박스 응용은 어떤 사람이 그 방에 입장하거나 떠날 때 그것이 재생하는 노래, 그 음악을 재생하는데 사용되는 볼륨 또는 스피커를 변경하여 스스로 재구성될 수 있다. 상황 프레임워크는 개발자에게 다른 상황에서 다른 행위를 명시하기 위한 다수의 방법들을 제공한다. 상황 합성자의 경우처럼, 그러한 행위를 기술할 수 있는 2가지 방법이 있다. 첫 번째는 응용 개발자가 다른 상황에서 무슨 행위가 수행되어야 하는지 가리키는 규칙을 작성하는 것을 허용하는 것이다. 두 번째는 다른 상황에서 무슨 행동이 수행할지 학습하는 기계 학습 방법을 사용하는 것이다. 응용 행위를 명시하기 위하여 규칙을 사용하는 것은 상황 센스티브 응용을 만드는 매우 간단한 방법이다. 그러한 규칙들은 조건과 행위로 구성된다. 환경의 상황이 바뀌면 모든 규칙에서의 조건들이 평가되어진다. 만일 어떤 조건이 참이 되면, 그러한 규칙들에 대응하는 행위가 수행되어 진다. 각 규칙은 그 행위들에서 충돌이 있을 경우에 사용되는 어떤 우선순위와 역시 관련되어 진다. 규칙에서 조건은 일차 논리, 시간 논리, 기술 논리, 고차 논리, 퍼지 논리 등과 같은 몇몇 형태로 된 표현이다. 상황인식을 위한 프레임워크는 규칙 작성을 위한 어떤 형태의 논리의 사용을 허용하기 위하여 충분히 유연해야 한다. 조건을 표현하기 위하여 사용되는 논리의 종류에 따라, 어떤 조건

이 참 또는 거짓인지 결정하기 위하여 다른 평가 엔진이 사용된다. 그 미들웨어는 규칙 기반 상황 소비자 개발을 쉽게 만든다. 그러한 에이전트들은 모든 규칙들을 나열한 어떤 구성 파일을 가지고 있다. 행위는 그 상황이 참일 때 호출되는 에이전트내의 메서드들로 명시된다. 그 미들웨어는 적절한 상황 제공자에 대한 참조를 획득하고, 그것들로부터 상황 정보를 획득하고, 규칙을 평가하고 다른 상황에서 적절한 메서드 호출을 위한 지원을 제공한다. 그것은 추론 과정에서 도움이 되는 라이브러리 형태의 다른 가용 평가 엔진을 역시 만든다. 에이전트 개발자의 작업은 매우 간단하다. 개발자는 다른 것에 대해 걱정할 필요 없이 에이전트 행위를 결정하는 규칙 작성에 집중할 수 있다.

주크박스 응용 에이전트를 위한 간단한 구성 파일은 표 4에서 보여준다. 이 에이전트는 방에 누가 있는가, 바깥 날씨가 어떤가 그리고 그 사용자의 주식 투자가 어떻게 되었는가에 따라 방에 적합한 음악을 재생한다. 이 에이전트의 규칙들은 일차 논리로 작성되었다. 상황 센스티브 응용 개발에서 규칙 기반 방법의 단점은 유연하지 못하고 환경 변화를 스스로 수용할 수 없다는 것이다. 기계 학습 기법의 사용은 이 문제 극복에 도움을 준다. 개발자는 다른 시나리오에서 응용의 행위를 명시할 필요가 없다. 응용은 다른 상황에서 가장 적절한 행위를 학습할 수 있다. 다양한 기계 학습 기법들이 적절한 행위를 학습하기 위하여 사용될 수 있다. 그러한 것들은 베이지안 방법, 신경망, Support Vector Machine, 다양한 클러스터링 알고리즘, 강화 학습 등을 포함한다. 학습은 일괄처리 또는 온라인 방식으로 발생할 수 있다. 일괄처리 방식은 다수의 훈련 예를 요구한다. Gaia는 환경에서 전송되는 모든 이벤트들을 데이터베이스에 저장한다. 그러한 것들은 상황 정보를 갖는 이벤트들, 사용자와 응용 행위를 기술하는 이벤트들을 포함한다. 저장된 과거 이벤트들은 학습자를 위한 훈련 예로서 역할을 한다. 이 방법은 어떤 주기 동안 사용자의 행위를 연구하여 사용자 행위를 학습하는데 특히 유용하다. 만일 사용자 행위가 잘 학습되었다면, 응용은 상황에 따라 그 사용자를 위한 사전 행위를 수행할 수 있고 사용자의 귀중한 시간을 절약할 수 있다.

(표 4) 방에서 활동 추론을 위한 일차 논리

```
ThereExists (Person) x Location (x, Entered, 2401).
PlayWelcomeMessage().
Priority:1.

Location(Manuel, Entered 2401) OR Location(Chris,
Enetred, 2401).
ShowInterface().
Priority:1.

Location(Manuel, In, 2401).
PlayRockMusic().
Priority:1.

Location(Manuel, In, 2401) AND
Temperature(Champagn, >, 50)
PlaySoftMusic()
Priority:2

Location(Bhaskar, In, 2401) AND Location(Chris, In,
2401)
PlayPopMusic()
Priority:2

Location(Bhasker, In, 2401) AND Location(Chris, In,
2401) AND Temperature(Champaign, >, 50)
PlayHiphopMusic()
Priority:4

Location(Bhasker, n, 2401) AND StockPrice(MSFT, >,
50)
PlayHappyMusic()
Priority: 2

Location(Bhasker In, 2401) AND StockPrice(MSFT, >,
20) AND StockPrice(SUNW, >, 10)
PlayVeryHappyMusic()
Priority:3
```

온라인 학습 메커니즘은 그 환경에서 작동 중에 그것들의 개념을 학습할 수 있다. 그러한 메커니즘은 다른 행위 시험, 그러한 행위에 대한 사용자의 반응 관찰 그리고 다른 상황에서 더 좋은 행위 학습을 포함한다. 예를 들어, 다른 종류의 통지를 전송하기 위한 가장 적절한 횟수를 학습하는 지능형 통지 서비스를 개발한다. 그것은 음성 또는 전자우편으로 주식 가격,

날씨 정보, 뉴스 헤드라인 그리고 오류 메시지와 같은 다른 종류의 통지를 전송할 수 있다. 그 통지 서비스는 몇몇 상황에서 어떤 통지를 전송할지 그리고 그 통지에 대한 사용자의 반응에 대한 관찰을 학습한다. 사용자는 그것의 유용성을 평가하여 통지에 대한 피드백을 줄 수 있다. 사용자의 피드백에 따라, 그 통지 서비스는 유사한 상황에서 같은 종류의 통지를 다시 전송할 확률을 증가시키거나 감소시킨다.

## 5.4 상황 제공자 검색 서비스

상황 제공자 검색 서비스는 다른 상황 제공자들을 위한 검색을 허용한다. 상황 제공자들은 상황 제공자 검색 서비스를 제공하는 상황 집합을 광고한다. 이 광고는 일차 표현식의 형식이다. 에이전트들은 그것이 필요한 상황 정보를 제공하는 상황 제공자에 대한 검색 서비스를 질의할 수 있다. 그 검색 서비스는 그 상황 제공자들 중의 어떤 것이 그 에이전트가 요구하는 것을 제공할 수 있고 그 응용에 결과를 반환할 수 있는지 검사한다. 예를 들어, 빌딩 주변의 Bob의 위치를 추적하는 어떤 위치 상황 제공자는 $\forall_{\text{Location}}$ y Location( Bob, In, y )로 자신을 광고한다. Bob이 방 3231에 들어갈 때를 알기를 원하는 어떤 응용은 검색 서비스에 질의 Location(Bob, In, Room 3231)을 전송할 것이다. 그 검색 서비스는 그 상황 제공자가 그 응용이 관심을 갖는 상황을 제공하는지와 그 응용을 위해 상황 제공자에게 참조를 반환하는지를 알고 있다.

## 5.5 상황 히스토리

응용은 사용자들과 더 좋은 상호작용을 위하여 그것들의 행위를 적응시키기 위하여 방금 현재 상황뿐만 아니라 과거 상황을 사용할 수 있다. 상황이 발생하면 데이터베이스에 상황을 계속해서 저장한다. Gaia 이벤트 서비스에서, 이벤트 채널로 전송된 모든 이벤트들은 그 이벤트가 전송된 때를 가리키는 타임스템프에 따라 데이터베이스에 저장된다. 어떤 데이터베이스에 모든 상황 이벤트를 저장하는 것이 가능하다. 모든 상황 이벤트들은 잘 결정된 구조를 가지므로, 그것들을 데이터베이스에 저장하기 위한 스키마를 자동적으로 개발하는 것은 상대적으로 간단하다. 과거 상황을 저장하는 것은 데이터 마이닝 사용이 사용자 행위에서 패턴, 방 활동 그리고 다른 상황을 학습하고 발견하는 것을 가능하게 한다. 이런 종류의 데이터 마이닝은 침입 탐지와 같은 보안 응용에서 사용될 수 있다. 여기서 정상에서 벗어난 어떤 관찰된 행위는 침입으로 간주된다.

## 5.6 시맨틱 상호운용성을 위한 온톨로지

미들웨어는 다양한 상황의 시맨틱을 정의하기 위하여 온톨로지를 사용한다. 온톨로지는 개념과 관계들로 구성된 사전으로서 특정 도메인에 관련된 객체들을 계층적 구조로 표현하고 추가적으로 이를 확장할 수 있는 추론 규칙을 포함한다. 유비쿼터스 환경에서는 모든 객체들과 객체의 속성들을 특정 종류의 도메인 안에 계층적으로 표현해야 하고, 이벤트 상황에 따라서 속성 값들을 변화시키는 규칙들이 필요하므로 온톨로지를 사용하여 유비쿼터스 환경내의 모든 객체들을 계층적으로 구성, 관리해야 한다.

유비쿼터스 컴퓨팅 환경은 다수의 자치적인 에이전트들로 특징 지워진다. 다양한 종류의 미들웨어는 다른 엔티티들 간의 통신이 가능하게 개발되었다. 그러나 기존의 미들웨어는 다른 엔티티들 간의 시맨틱 상호운용성을 보장하는 기능이 없다. 에이전트들이 자치적이므로, 그것들의 전부가 독자적인 다른 개념에 같은 시맨틱 부착을 기대할 수 없다. 다른 에이전트는 현재 상황의 다른 이해를 가질 수 있고 상황을 설명하기 위하여 다른 용어와 개념을 사용할 수 있기 때문에 이것은 상황 정보에 대해 특히 그렇다. 다른 에이전트들 간의 시맨틱 상호운용성을 가능하게 하기 위하여, 시맨틱 웹에서 사용되는 방법에 의지한다. 온톨로지는 관심 커뮤니티의 멤버들 간의 공통 용어를 설립한다. 그러한 멤버들은 사람이거나 자동화된 에이전트일 수 있다. 유비쿼터스 환경에서 각 에이전트는 하나 이상의 온톨로지에서 정의된 어휘와 개념을 사용한다. 2개의 다른 에이전트가 서로 말할 때, 그것들은 다른 에이전트가 사용하는 온톨로지를 알고 다른

에이전트가 무엇을 말하는지 시맨틱을 이해할 수 있다. 시맨틱을 기술하기 위하여 시맨틱 웹에서 사용되는 표준 기술을 사용하는 다른 장점은 확장성이다. 유비쿼터스 환경에서 에이전트들은 웹에서와 같은 언어(DAML+OIL)로 기술된 온톨로지를 사용하므로, 상황인식 에이전트와 외부 에이전트 간의 시맨틱 상호운용성을 준다. 온톨로지의 사용은 다른 유비쿼터스 환경에서 에이전트들이 서로 상호작용할 때 공통 어휘와 개념의 공통집합을 가지는 것을 가능하게 한다.

Gaia에서 모든 온톨로지는 온톨로지 서버에 의해 관리된다. Gaia에서 다른 에이전트들은 그 환경에서 에이전트들의 기술, 상황에 대한 메타 정보 또는 Gaia에서 사용되는 다양한 용어를 얻기 위하여 온톨로지 서버와 접촉한다. 그것은 시맨틱 질의를 지원할 수 있다. 온톨로지 서버는 기존의 온톨로지에 새로운 개념을 추가하기 위하여 인터페이스를 역시 제공한다. 이것은 새로운 종류의 상황이 어떤 시점에 그 환경에 도입되고 사용되어지는 것을 허용한다. 그 온톨로지 서버는 어떤 새로운 정의가 기존 정의와 논리적으로 일관성을 유지한다. 온톨로지의 사용은 다른 환경에서 에이전트들이 상호 작동하는 것을 가능하게 하였다. 그러한 상호 작동을 지원하기 위하여, 사상(mapping)이 2개 환경의 온톨로지에서 정의된 개념들 사이에 개발되어야 한다.

온톨로지는 상황 정보의 구조를 분명히 정의하므로, 다른 에이전트들은 다른 종류의 상황 정보를 쉽게 교환할 수 있다. 상황 소비자 에이전트들은 온톨로지 서버로부터 그것들이 관심이 있는 상황의 구조를 얻을 수 있다. 상황 소비자 에이전트들은 그것들이 이벤트 채널로 보내야 하는 이벤트의 구조를 역시 안다. 결국, 개발자가 상황 인식 에이전트를 위하여 규칙을 작성하거나 학습 메커니즘을 개발할 때 온톨로지는 개발자에서 역시 도움을 준다. 개발자는 상황 정보를 기술하는 용어와 개념의 집합에 대한 액세스를 갖는다. 개발자는 상황 인식 에이전트를 개발할 동안 가장 적절한 용어와 개념을 사용할 수 있다.

## 6. 결론 및 향후연구

이 논문에서, 우리는 상황인식 응용 개발을 위한 미들웨어에 대하여 설명하였다. 이 미들웨어는 상황 술어 모델에 기초한다. 이 모델은 에이전트가 다른 상황에서 다른 행위를 결정하기 위하여 규칙이나 기계 학습 방법을 사용하여 개발되어지는 것을 가능하게 한다. 그 미들웨어는 그 환경내의 다른 에이전트들이 다른 상황 정보에 대한 동일한 이해를 가지는 것을 보장하기 위하여 온톨로지를 사용한다. 이것은 다른 에이전트들뿐만 아니라 다른 유비쿼터스 컴퓨팅 환경들 간의 더 좋은 시맨틱 상호운용성을 제공한다. 이러한 미들웨어는 상황 센스티브 응용의 빠른 원형 생성을 허용하고 그 미들웨어 위에 다수의 상황 센스티브 에이전트들을 아주 쉽게 개발할 수 있다. 앞으로 논문에서 소개한 미들웨어에 보호 및 보안 기능의 추가, 상황인식 응용 개발을 위한 더 편리한 사용자 인터페이스 개발 등에 관한 연구가 필요할 것으로 사료된다.

## 참 고 문 헌


[1] A. Dey and D. Salber and G. Abowd, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” Human-Computer Interaction(HCI), Vol. 16, 2001.

[2] J. Hong and J. Landay, "An Infrastructure Approach to Context-Aware Computing," HCI Journal, Vol. 16, 2001.

[3] P. Bouquet and P. Busetta and M. Bonifacio, "Context Aware Distributed Applications," IRST Techical Report 0101-04, Instituto Trentino di Cultura, January 2001.

[4] Paul Castro, Murali Mani, Siddhartha Mathur and Richard R. Muntz, "Managing Context for Internet Video Conferences: The Multimedia Internet Recorder and Archive," Proc. of Multimedia and Computer Networks, San Jose, California, 2000.

[5] Winograd, Terry, "Architectures for Context," Human-Computer Interaction, Vol. 16, 2001.

[6] Steven A. N. Shafer, Barry Brumitt, JJ Cadiz, "Interaction Issues in Context-Aware Intelligent Environments," Special issues on Context-Aware Computing, Human-Computer Interaction (HCI)

Journal, Vol.16, 2001.

[7] H. Chen, "An intelligent broker architecture for context-aware systems," PhD. Dissertation, University of Maryland, 2003.

[8] Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," In IEEE Pervasive Computing, pp. 74-83, 2002.

[9] T. Gu and H. Pung and D. Zhang, ""A Middleware for Building Context-Aware Mobile Services", In Proceedings of IEEE Vehicular Technology Conference, Milan, Italy, May 2004.

[10] Marcela Rodriguez and Jesus Favela, "A Framework for Supporting Autonomous Agents in Ubiquitous Computing Environments," UbiSys '03, 2003.

[11] Jason Pascoe, Nick Ryan, David Morse, "Issues in Developing Context-Aware Computing," Lecture Notes in Computer Science, Volume 1707, 1999.

[12] William Noah Schilit, "System Architecture for Context-Aware Mobile Computing," PhD. Dissertation, Columbia University, New York, May 1995.

[13] Nicola Guarino, "Formal Ontology and Information Systems," Proceedings of FOIS '98, Trento, Italy, pp. 3-15, 1998.

[14] Frank van Harmelen, Peter F. Patel-Schneider and Ian Horrocks, "Reference description of the DAML+OIL ontology markup language," http://www.daml.org/2001/03/ reference.html

[15] Anand Ranganathan, Roy H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," Lecture Notes in Computer Science, Volume 2672, 2003.

## ◐ 저 자 소 개 ◑



**배 인 한**

- 1984년 경남대학교 전자계산학과(공학사)
- 1986년 중앙대학교 대학원 전자계산학과(이학석사)
- 1990년 중앙대학교 대학원 전자계산학과(공학박사)
- 1996년~1997년 Department of Computer Science and Engineering, The Ohio State University(Post- Doc)
- 2002년~2003년 Department of Computer Science, Old Dominion University (Visiting Professor)
- 1989년~현재 대구가톨릭대학교 컴퓨터정보통신공학부 교수
- 관심분야 : 모바일 멀티미디어, 모바일 컨버전스, 모바일 컴퓨팅, 무선 인터넷 등