# 상황 인식 응용을 위한 OSGi 기반 서비스 미들웨어

정 헌 만[†] · 이 정 현[††]


## 요 약

유비쿼터스 컴퓨팅 환경에서 상황 인식 기반의 서비스를 제공하기 위해선 동적인 상황 관리 기술과 상황 추론 기술, 그리고 상황 모델링 기술이 필요하다. 기존 연구에서 상황 인식 서비스는 상황 인식 미들웨어에서 사용하는 상황 온톨로지를 이용해서 설계되기 때문에 서비스의 실행도중 상황 온톨로지를 동적으로 변경하기 어렵다. 본 논문에서는 유비쿼터스 컴퓨팅 환경에서 상황 변화에 대한 사용자의 요구사항을 능동적으로 반영하고, 동적인 응용 적응성을 지원하는 계층적 온톨로지 기반 상황 관리 모델을 제안하고 이를 기반으로 상황 인식 미들웨어를 설계한다. 또한, 상황 인식 서비스 구현을 위해 다양한 컨텍스트 발견, 획득, 해석, 추론을 효과적으로 지원하며 사용자의 서비스 실행 시 발생할 수 있는 상황 충돌을 해결하기 위한 방법을 제시한다. 설계하는 미들웨어는 OSGi 프레임워크 위에서 구현함으로써 다양한 유비쿼터스 환경에 필요한 상황 인식 서비스의 개발 및 운용을 효과적으로 지원을 할 수 있다.


**키워드 : 상황인식 모델, 상황 온톨로지, 상황 인식 미들웨어, OSGi, 유비쿼터스 컴퓨팅**

# OSGi based Service Middleware for Context-Aware Applications

Heon-Man Jung[†] · Jung-Hyun Lee[††]


## ABSTRACT

To support context-aware services in ubiquitous computing environments, there are required dynamic context managing, context reasoning and context modeling technologies. In previous researches, context services are designed using context ontology used in context aware middleware. So, context service cannot change the context ontology in execution time. In this paper, we propose a hierarchical ontology-based context management model and design a context-aware middleware based on this model for supporting active application adaptability and reflecting users' requirements dynamically in contextual changes. It also provides efficient support for inferencing, interpreting, acquiring and discovering various contexts to build context-aware services and presents a resolution method for context conflict which is occurred in execution of service. As the middleware is implemented on the OSGi framework, it can cause interoperability among devices such as computers, PDAs, home appliances and sensors. It can also support the development and operation of context aware services, which are required in the ubiquitous computing environment.


**Key Words : Context-Aware Model, Context Ontology, Context-Aware Middleware, Open Services Gateway Initiative, Ubiquitous Computing**

## 1. 서 론

유비쿼터스 컴퓨팅은 일상생활 속에 편재해 있는 컴퓨팅 자원을 이용하여 사용자가 언제 어디서나 동적인 서비스를 받을 수 있는 환경을 제공하며 조용한 컴퓨팅(calm computing)[1], 보이지 않는 컴퓨팅(invisible computing)[2], 사라지는 컴퓨팅(disappearing computing)[3]등의 용어는 유비쿼터스 컴퓨팅에 관한 사용자 인터페이스 관점을 잘 설명해 주고 있다.

유비쿼터스 환경에서는 기존 컴퓨팅 환경에서의 사용자와 컴퓨터간의 대화형 상호작용이 아닌 물리적인 환경, 상황(context)등을 시스템이 인식하고 이를 기반으로 사용자와의 상호 작용을 지원하는 상황 인식 기술이 필수적인 요소로 자리 잡고 있다. 또한 상황 인식 서비스는 다양한 상황 정보를 수집 및 해석을 통해 인식하고, 추론과정을 거쳐 기존의 수동적인 서비스에서 벗어나 사용자 명령 없이도 자동으로 실행되는 지능형 서비스를 지원하며, 각 사용자에게 맞춰진 개인화된 서비스 등을 제공한다. 상황 인식 서비스는 의료, 교육, 재난, 구호, 쇼핑 등 사회 전 분야에 걸쳐 응용될 수 있어 많은 영향을 줄 것이다[4, 5].

기존의 상황 인식 서비스 기반 기술 중 상황에 대한 체계적 관리를 위한 온톨로지 기술과 사용자 맞춤형 서비스 제공을 위한 추론 기술에 대한 연구는 어느 정도 진행되고 있

† 정 회 원 : 경인여대, 인하전문대 강사
†† 종신회원 : 인하대학교 컴퓨터공학부 교수
논문접수 : 2006년 7월 26일, 심사완료 : 2006년 9월 7일

으나 사용자 사이에 발생하는 서비스 충돌 해결에 대한 연구는 부족한 실정이다[4, 5, 6, 7].

본 논문에서는 상황 인식 모델링 및 센싱에 필요한 기반 기술과 온톨로지 기반 기존 연구에 대해 알아보고 OSGi기반의 상황 인식 미들웨어를 설계한다. 제안한 모델은 사용자의 행동 및 환경의 변화에 따른 센서 입력 데이터를 바탕으로 온톨로지를 구성하고 센서로부터 입력되는 다양한 종류의 상황을 좀 더 자연스럽게 처리하기 위해 입력되는 상황의 타입(이산적, 연속적) 및 확률 정보를 온톨로지 내에 표현하여 추론할 수 있도록 온톨로지 모델을 설계한다. 또한 센서로부터 얻은 데이터가 2개 이상의 상황 정보로 해석되는 상황 의 모호성 문제를 해결할 수 있는 방법을 제안하고 OSGi을 지원함으로써 이기종간의 통합 및 상호 운용성을 효과적으로 지원할 수 있게 한다.

## 2. 관련 연구

이 장에서는 유비쿼터스 컴퓨팅에서 필수적으로 요구되는 상황 인식 관련 기술과 모델링 방법, 상황 인식에 관련된 기존 연구 및 OSGi에 대해서 기술한다.

### 2.1 상황 인식 개요

상황에 대한 다양한 정의가 있지만 사용자와 유비쿼터스 컴퓨팅 환경 사이의 관계와 연관 지어지는 사용자 주위의 상황이나 상태(circumstance) 또는 객체(object)들에 대한 정보를 통칭한다[4]. 상황의 본질적인 정의는 "실세계(real world)에 존재하는 실체(entity)의 상태를 특징화하여 정의한 정보"라고 정의할 수 있으며, 여기서 실체란 인간, 장소 또는 사람과 서비스간의 상호 작용을 의미한다고 할 수 있다.

Schilit와 Theimer는 그들의 연구에서 "상황 인식"이란 용어를 처음으로 사용하였는데, 그 연구에서는 상황을 장소, 사람이나 사물들을 구별 짓는 특징인 아이덴티티(identity), 사람이나 사물들을 포함하는 환경의 변화 등으로 설명한다[5]. Dey는 상황을 사용자가 속해 있는 환경 내에서 사용자의 감정적인 상태, 주의력, 위치와 방향, 날짜와 시간, 사람과 사물 등으로 정의한다[6].

위와 같이 관점에 따른 상황 정의에 약간의 차이가 있으나 일반적인 상황 정보는 사용자 상황, 물리적 환경 상황, 컴퓨팅 시스템 상황, 사용자-컴퓨터 상호 작용 이력, 기타 상황으로 분류할 수 있으며 사용자의 현재 상황에 따라 적절한 정보 혹은 서비스를 제공하기 위해 상황을 이용하는 것을 상황인식(context awareness)이라 한다[4, 7].

상황 모델링은 상황 정보에 대한 높은 수준의 추상적 개념을 제공하기 위해 필요하며 센서와 엑추에이터(actuator)의 추상화를 제공함으로써, 개발자가 다양한 하드웨어 장치와 인터페이스하는 부담을 줄일 수 있다. 이러한 상황 모델링 기법은 각 시스템에서의 상황 정보 교환을 위해 사용되는 데이터 구조에 관한 스키마에 의해 다음의 네 가지로 분류된다[4, 5, 6, 7].

키-값(key-value) 모델은 상황 정보를 모델링하기 위한 가장 간단한 데이터 구조이다. 환경 변수로서 상황 정보(예, 위치 정보)의 수치(value)를 어플리케이션에게 제공하는데, 키-값 쌍의 형식을 사용하여 상황을 모델링하며 Key-Value 모델은 관리하기 쉽지만, 효과적인 상황 검색 알고리즘을 위한 복잡한 구조화가 부족하다.

마크업 스키마(markup scheme) 모델링 기법은 속성과 내용을 갖는 마크업 태그로 구성된 계층적 데이터 구조이다. 특히, 마크업 태그의 내용은 항상 다른 마크업 태그에 의해 재귀적으로 정의될 수 있다. 이러한 종류의 상황 모델링 기법에서는 전형적으로 프로파일이 대표적이다.

객체 지향 모델링 기법은 유비쿼터스 환경에서 상황의 동적 특성에 관한 문제점들을 해결하기 위해서, 캡슐화와 재사용이라는 객체 지향의 주요 장점을 적용한다. 상황 처리에 관한 자세한 내용은 객체 수준에서 캡슐화 되므로, 다른 컴포넌트들에게 은닉되고 상황 정보로의 접근은 특정 인터페이스를 통해서만 제공된다.

온톨로지 기반 모델링은 온톨로지는 개념과 상관관계를 기술하는 도구로 상황정보를 표현하고 공유하기 위한 어휘 및 용어를 제공하며 상황 정보의 다양성 때문에 도메인 기반 온톨로지(domain specific ontology)를 정의하고 있다. 이 모델은 유비쿼터스 컴퓨팅 환경에서 상황 지식의 공유와 재사용을 지원하며, 또한 계층적 상황 온톨로지 모델에서는 상위 계층의 온톨로지를 이용하여 도메인에 맞는 하위 계층의 온톨로지를 생성해 낼 수 있다.

### 2.2 상황 인식 서비스 미들웨어

현재 상황 인식과 관련된 연구는 상황 인식 미들웨어와 미들웨어를 이용한 서비스 즉 응용으로 분류할 수 있으며 이 절에서는 관련된 대표적인 연구에 대해 요약하였다.

Gaia[8]는 응용이 다양한 상황정보를 얻고 추론할 수 있게 해주며, 상황 처리를 위해 논리 추론과 기계 학습 방법이 폭넓게 활용되며, 서로 다른 유비쿼터스 컴퓨팅 환경뿐만 아니라 이종 에이전트간의 시맨틱한 상호 운용성을 보장하기 위해서 DAML(Darpa Agent Markup Language)+OIL로 기술된 온톨로지를 사용한다.

SOCAM(Service Oriented Context-aware Middleware)[9, 10]은 유비쿼터스 환경에서 제공 되어지는 다양한 상황의 상호의존적 개념에 대한 정의 및 모바일 환경에서의 상황인식 서비스 제공을 위해 구현되었으며 OWL[11]기반의 온톨로지를 사용하여 의미기반 상황 표현과 다양한 형태의 상황에 대한 추론, 지식 공유, 상황의 분류와 상호 의존성과 관련된 문제를 다루고 있다. SOCAM은 OSGi를 기반으로 Java로 구현되었으며, 상황 해석을 Jena2를 이용하였다.

DyCAM(Dynamic Context Aware Middleware)[12]에서는 유비쿼터스 환경에서 서비스의 이동성을 효과적으로 지원하기 위해서 상황 정보의 동적 관리 및 서비스간의 상호 작용을 위한 서비스 검색 및 조합, 서비스 이동성을 지원하는 모델 및 미들웨어를 제시했으며 OSGi 프레임워크위에서

구현함으로써 UPnP, Jini등의 표준 인터페이스 기술을 이용할 수 있도록 했다.

상황 인식 응용 서비스로 MicroSoft의 Easy Living[13, 14]은 사용자 신원, 위치, 대상물 인식 정보를 이용하여 정보 가전기기를 제어하는 지능형 가정환경을 구축한다. 객체와 사람, 공간사이의 물리적 관계의 정보를 상황정보화 하여 사용함으로써 상황을 추론하고 사용자가 원하는 서비스를 제공한다.

Georgia Tech. Aware Home Research Initiative(AHRI)의 AwareHome[15, 16]은 가정 내 사용자의 상황(누가, 어디에서, 무엇)을 파악하는 연구로 실내에서의 위치 인식을 위해 RFID와 마루 메트를 이용했으며, 혼자 사는 노인들의 위치 정보와 활동 상태 정보를 파악하여 병원이나 보호자에게 알려주는 스마트 홈의 모델을 제시하였다

Musex[17]는 박물관에서 관람하고 있는 어린이를 대상으로 실시한, 학습자 위치 인식에 맞춰 학습자 부근의 전시물과 관련된 컨텐츠를 RFID가 부착된 PDA(Personal Digital Assistants)에게 퀴즈 형태로 제공하고 사용자의 흥미를 유발시키도록 하는 연구이다.

### 2.3 OSGi(Open Services Gateway Initiative)

OSGi는 홈 네트워킹, 텔레매틱스, 산업 자동화 등의 분야에서부터 기타 정보 가전 및 다양한 개인 단말 등과 같은 환경 내에서 각종 서비스를 배포, 관리할 수 있게 해주는 개방형의 서비스 게이트웨이 아키텍처를 개발하고 표준화시키기 위한 국제표준화 단체이다. 초기에는 홈서비스 게이트웨이에 집중되었지만 최근에는 특정 네트워크 환경에 국한하지 않고 유비쿼터스 환경까지 확장해 가고 있다. 따라서, 네트워크로 연결되는 다양한 임베디드 디바이스와 이를 이용하는 사용자에게 서비스를 제공하기 위한 게이트웨이 구축을 목표로 하고 있다[18].

OSGi는 OSGi 서비스 플랫폼을 위한 스펙을 정의하며 플랫폼은 OSGi 프레임워크와 기본 서비스 정의들로 구성되는데, OSGi 프레임워크는 서비스의 배치 및 실행 환경인 서비스 게이트웨이를 정의한다. 실제로 OSGi 프레임워크는 서비스 지향의 어플리케이션들을 실행하고 배치하며, 서비스 레지스트리 및 컴포넌트 모델을 지원한다[19].

OSGi 프레임워크는 서비스들을 위한 실행 환경을 의미하며 최소한의 컴포넌트 모델, 컴포넌트를 위한 관리 서비스, 서비스 레지스트리 등을 포함한다. OSGi 프레임워크는 번들이라고 불리는 OSGi 컴포넌트를 설치하고, 서비스 등록 및 실행을 위한 프로그래밍 모델을 지원한다. OSGi 프레임워크 자신도 번들로 표현되며, 이러한 번들을 시스템 번들이라고 한다.

번들은 서비스 레지스트리에 등록된 서비스를 이용하는 서비스 집합인 동시에 컴포넌트 단위이다. 서비스 구현은 물리적이며 논리적인 단위인 번들을 통해서 프레임워크에 전달되고 배치된다. 물리적으로, 번들은 코드, 리소스 그리고 프레임워크에게 번들 클래스의 실행 경로를 알려주고 다른 번들과 공유하게 될 자바 패키지를 선언하는 Manifest 파일 등을 포함하는 자바 Archive(.jar) 파일 형태로 배포된다.

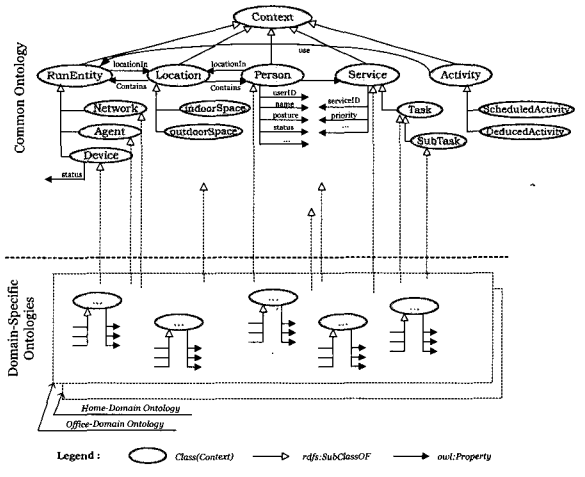
현재 상황 인식 서비스에 대한 다양한 연구가 진행되고 있으나 OSGi 프레임워크 기반 UPnP, Jini등의 표준 인터페이스 기술을 이용하여 다양한 가전과 센서 등 이기종과의 상호 운용성이 가능한 미들웨어 대한 연구가 부족하고 사용자간에 발생하는 서비스 충돌 해결 및 불확실하고 모호한 상황 데이터의 처리에 대한 연구가 계속되고 있다.

## 3. 상황 인식 모델

온톨로지를 기반으로 상황을 모델링함으로써 쉬운 추론 및 패턴 분석, 사람과 기계, 기계사이의 원활한 커뮤니케이션, 확장성 및 상황정보 공유의 용이성, 다양한 응용 서비스 관리의 편리함을 얻을 수 있다. 그러나 현재 상황인식 시스템에서 사용되는 상황 정보 값은 실세계의 값과 일치하지 않아 자연스러운 서비스 제공에는 한계가 있다. 본 모델에서는 상황 데이터에 대한 확률 추론을 위해 기존의 온톨로지 모델에 확률 표기를 추가한 온톨로지 모델을 제안하며 상황 온톨로지 모델을 계층적으로 구성하여 상위 계층의 공통 온톨로지를 상속받아 특정 도메인에 적합한 하위 계층의 온톨로지를 생성한다.

### 3.1 계층적 상황 온톨로지 구성

(그림 1)은 이 논문에서 설계한 계층적 상황 온톨로지를 표현한 것이다. 미들웨어 상에서 관리되는 상황 정보는 상위 계층의 공통 온톨로지 상황 정보와 도메인별 특성에 맞게 상위 온톨로지를 상속하여 별도로 정의되는 하위 도메인 상황 정보로 구성되며, 상황 및 속성은 웹 온톨로지 언어인 OWL로 온톨로지를 정의한다. 서비스가 미들웨어에 새롭게 배치되는 경우에는 미들웨어는 기존의 도메인 상황 정보를 상속받아 서비스에서 필요로 하는 개별 상황 정보를 추가하여 운용한다.




(그림 1) 계층적 상황 온톨로지 구성

공통된 상위 상황 정보는 상황 인식 응용에서 필요로 하는 기본 요소를 최상위 클래스인 Context 클래스에서 상속받아 실행객체(RunEntity), 위치(Location), 사람(Person), 서비스(Service), 행위(Activity) 클래스로 정의하고 도메인별 상황 정보는 상위 클래스의 정보를 상속받아서 설계한다.

각 상황 클래스는 해당 상황을 설명하기 위한 다수의 속성을 포함할 수 있으며, 이 속성은 다른 클래스와의 관계(relation)를 나타낼 때도 사용되며, 다른 상황을 프로퍼티로 포함할 수도 있다. 예를 들어서, 홈 네트워크 응용 온톨로지에서 Location 클래스로부터 상속받은 indoorSpace의 BedRoom 상황과 RunEntity의 하위 클래스인 Device 클래스로부터 상속받은 Temperature 상황이 있는 경우 Temperature 상황은 Room 상황의 프로퍼티로 이용될 수 있다.

### 3.2. 상황 정보의 구성

상황 인식 시스템에서 사용되는 상황 정보는 아래 <표 1>과 같이 구성될 수 있으며 다양한 형태로 입력되는 상황 데이터의 전처리를 위해서 필터링과 이산화 단계를 거친다. 기존 시스템의 센서데이터는 대부분이 켜짐/꺼짐과 같은 1비트로 표현되는 경우가 많아 데이터 필터링 단계에서는 전처리가 필요하지 않지만 본 모델에서는 이산적인 상황 데이터를 2비트이상의 정보로 표현하며, 연속적인 데이터 값인 사람의 위치, 시간, 장소, 온도등과 같은 경우 전처리 과정을 통해 관측된 데이터를 미리 정의된 퍼지 소속도 함수를 사용하여 각 상태에 대한 퍼지 소속도 값을 계산하여 0과 1사이의 실수 값으로 정의 하며, 계산 양이 적으면서 비교적 성능이 좋은 사다리꼴 모양의 퍼지 소속도 함수를 사용한다.

상황 정보는 <표 1>과 같이 연속적인 데이터와 이산적인 데이터로 구분하여 데이터 타입에 따라 적절한 전처리과정이 진행될 수 있도록 한다. 또한 실세계와 유사한 상황을 유지하기 위해서 이산적인 상황 데이터를 2비트 이상으로 표현된다.

〈표 1〉 상황 정보의 구성

| Cotext Data | Acquire Method | Data Type | Context Value | Info Bits |
|---|---|---|---|---|
| User | RFID | object | userID | |
| Activity | user,object | object | service | |
| Time | internal | continuos | time | |
| Season | internal | discrete,symbolic | spring,summer fall,winter | 4 |
| Location | RFID,GPS | continuos | value | |
| Bright | sensor | continuos | value | |
| Temperature | sensor | continuos | value | |
| Humidity | sensor | continuos | value | |
| Sound | sensor | continuos | value | |
| Noise | sensor | continuos | value | |
| Light | sensor | discrete,symbolic | Dark,Dim,Bright veryBright | 2 |
| Weather | external | discrete,symbolic | sunny,cloudy rain,snow | 3 |
| Window | sensor | discrete,symbolic | open,half,closed | 2 |
| Door | sensor | discrete,symbolic | open,half,closed | 2 |

아래 코드는 프로퍼티 'dataType' 속성을 정의하여 온도 센서를 연속적인 데이터 소스로 선언한 OWL 표현이다.

```
<owl:Class rdf:ID="Sensor">
  <rdfs:subClassOf rdf:resource="#Device"/>
</owl:Class>
<owl:Class rdf:ID="DefinedType">
   <inha:DefinedType rdf:about="#Continuous">
   <inha:DefinedType rdf:about="#Discrete">
   <inha:DefinedType rdf:about="#Symbolic">
</owl:Class>
<owl:ObjectProperty rdf:ID="dataType">
   <rdf:domain rdf:resource="#Sensor"/>
   <rdf:range rdf:resource="#DefinedType"/>
</owl:ObjectProperty>
<inha:Sensor rdf:ID="BedRoom_Temperature">
   <inha:dataType rdf:resource="#Continuous"/>
   <inha:hasPValue>float</inha:hasPValue>
</inha:Sensor>
```

### 3.3 온톨로지 기반 상황 모델

상황 인식 응용에서 사용되는 각각의 상황들은 일반적으로 사용자 및 주변 환경의 다양한 센서를 통해서 받은 원시 데이터를 통한 방법을 사용한다. 이렇게 얻어진 원시 데이터를 통해서 기본적인 상황 정보를 만들어내고, 다양한 추론을 통해서 보다 고차원적인 복합 상황을 유추하게 된다. 따라서 여러 센서들로부터 전달받은 원시데이터에서 기본 상황을 만들어내기 위한 모델이 필요하며, 이 모델은 센서의 특성상 잘못 전달된 값이나 오류에 대한 적절한 필터링 방법을 이용해야 한다. 이를 위해서 상황 정보를 파악하고 필터링하기 위한 기본 모델을 제안한다.

#### 3.3.1 상황

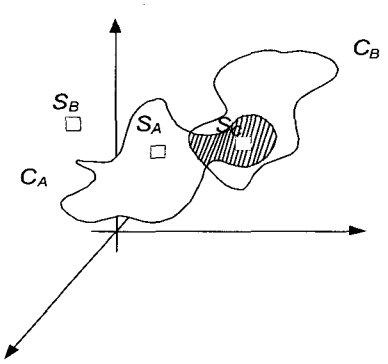
상황은 특정 시간에 센서로부터 수집된 여러 속성들이 일정 영역에 포함되는지에 따라서 결정되므로 시스템은 센서로부터 받은 입력 데이터 값을 통해서 상황을 판단할 수 있는 방법을 가져야 한다. (식 1)은 이 논문에서 설계하는 미들웨어에 연결되어 있는 센서들로부터 특정시간 (t)에 전달받은 속성들의 집합을 표현한 것이다.

$$S(t) = \sum_{i=1}^{n} A_i(t) \qquad \text{(식 1)}$$

(식 1)에서 Ai(t)는 특정 시간 t에 속성 값을 나타낸 것이다. 상황 속성은 상황을 추론하는데 사용되어질 수 있는 요소들을 기술하는데 사용된다. 상황 속성은 가상 또는 물리적인 센서들과 연결되어 있다.

#### 3.3.2 상황 공간

상황 공간은 특정 상황이 유추될 수 있는 속성들의 허용




(그림 2) 중첩되는 상황 공간

가능한 값의 범위로 정의한다. 상황 공간은 미리 정의된 상황에 대응되기 위한 센서로부터 얻어지는 속성 값들의 영역으로 표현된다. (식 2)는 상황 공간을 표현한 것이다.

$$C_i = \sum_{i=1}^{n} S_i \qquad S_i = \{A \mid P(A)\} \qquad \text{(식 2)}$$

P는 센서로부터의 속성 값이 허용가능한지를 판단하는 Predicate 함수이며, Ai는 함수 P를 만족하는 요소들의 집합으로 정의된다. 상황(C)는 센서로부터 받은 속성 값의 일정 범위(A)들의 합으로 표현된다. 센서에서 전달받은 값의 허용범위인 Ai는 특정 값들의 집합으로 구성된다. C(t)를 통해서 속성 값들의 허용 가능한 값들과 상황 정보를 연결시키고 있지만, 센서로부터 전달받은 속성 값들을 통해서 바로 특정 상황정보로 매핑 되지는 못한다. 왜냐하면, 상황 A를 나타내는 $C_A$와 상황 B를 나타내는 $C_B$가 서로 겹쳐지는 부분이 존재할 수 있기 때문이다. (그림 2)는 두 상황이 겹쳐지는 상황을 표현한 것이다.

$S_A$는 $C_A$에 포함되기 때문에 상황 A로 매핑 되고 $S_B$는 $C_A$, $C_B$ 어느 곳에도 포함되지 않기 때문에 어떠한 상황에도 매핑 되지 못한다. 반면에 $S_C$는 $C_A$와 $C_B$의 중첩되는 부분에 위치하고 있다. 이런 경우에는 센서로부터 인식된 속성 값들이 하나 이상의 상황 공간에 위치되기 때문에 어떤 상황 공간에 속하는지를 판단할 수 있는 필터링 과정이 필요하다.

이 경우 교차연산 연산자는 두 상황 공간간의 동일한 속성의 값을 갖는 공통의 영역을 포함하는 새로운 상황 공간을 만들어낸다. 두 상황공간의 교차연산에 포함되는 개체들은 다음의 (식 3)과 같이 정의할 수 있다. A는 조건 함수 P를 만족하는 속성들의 집합이고, B는 조건 함수 Q를 만족하는 속성들의 집합일 때, 두 상황공간간의 교차연산에 포함되는 개체는 조건 함수 P와 Q를 모두 만족하는 속성들로 표현된다.

$$A = \{V \mid P(V)\} \qquad B = \{V \mid Q(V)\}$$
$$a \in (A \cap B) \qquad iff \qquad P(a) \wedge Q(a) \qquad \text{(식 3)}$$

센서로부터 전달받은 속성 값들의 집합이 (식 3)과 같이 여러 상황 공간 내에 포함되는 경우에는 포함되는 상황공간과의 비교를 통해서 적절한 상황을 유추한다. 이를 위해서 이 논문에서는 속성 값들과 상황공간과의 차이를 판단하기 위한 상황 공간 편차 함수를 정의한다. 이 상황 공간 편차 함수를 통해서 편차 값이 작은 상황 공간으로 매핑한다. 상황 공간 편차 함수는 (식 4)와 같이 정의한다.

$$Context - Space\ Derivation = \sqrt{\sum_{i=1}^{n} (A_s^i - A_r^i)^2}$$
$$A_s : Attibutes\ from\ Sensors\ ,$$
$$A_r : \frac{MaxValue(A_r) + MinValue(A_r)}{2} \qquad \text{(식 4)}$$

3.3.3 확률 확장

OWL은 확률 정보의 표현을 지원하지 않기 때문에 확률을 이용한 상황 정보 표현 및 추론을 위해 온톨로지 기반 모델에서 확률 정보를 가질 수 있는 클래스와 프로퍼티를 정의하고 각 클래스의 프로퍼티 간의 인과관계(casual realtionship)를 기반으로 인스턴스간의 의존 관계를 구성한다. 이러한 클래스의 프로퍼티 사이의 인과관계는 dependOn 속성으로 나타낸다.

클래스의 프로퍼티들 사이의 확률과 종속성을 표기하기 위해 마크업 요소를 다음과 같이 정의한다.

〈표 2〉 확률 확장 마크업 요소

| Class | Name | Type | Description |
|---|---|---|---|
| Prob | hasPVariable<br>hasPValue | Object Property<br>Vaue Property | 확률 값을 갖는 노드명칭<br>확률 값 |
| All | dependOn | Object Property | 다른 propety와의 의존성 표현 |
| | prob(nTriple) | Vaue Property | Class의 Predicate에 확률 값을 표현 |

Prob는 확률 정보를 가지고 있는 클래스로 프로퍼티와 제약사항(restriction)를 가지고 있는 일반적인 클래스와 유사하다. hasPVariable는 확률 정보를 가지고 있는 프로퍼티이며 hasPValue를 사용하여 확률 값을 지정한다. Prob (Predicate(subject,value))는 각 각의 class의 predicate에 확률 값을 표현하고 확률 값은 0과 1사이의 값을 가지며 어떤 상황정보(sensed contexts, defined contexts, derived contexts)에도 적용될 수 있다. Prob(actStatus(Jung, WatchingTV))=0.7은 Jung이 현재 TV를 보고 있을 확률이 0.7라는 것을 의미한다.

온톨로지 내에 RDF triples 형태의 클래스 A의 확률 P(A)는 클래스 Prob의 인스턴스로 정의되며, P(A)=0.8의 확률적 마크업을 사용한 OWL 표현은 다음과 같다.

```
<inha:Prob rdf:ID="P(A)">
<inha:hasPVariable>
```

```
    <rdf:value>A</rdf:value>
  </inha:hasPVariable>
  <inha:hasPValue>0.8</inha:hasPValue>
  </inha:PriorProb>
```

dependOn은 종속성 정보를 마크업하며 종속성은 상황 정보의 중요한 특성으로 한 객체의 프로퍼티와 다른 객체와의 종속성으로 나타내며 종속 관계에 있는 객체의 상황 값에 의해 확률적인 상황 추론이 수행된다.

```
<owl:Class rdf:ID="WatchingTV">
  <rdfs:subClassOf rdf:resource="#Activity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="actStatus">
   <rdf:domain rdf:resource="#Person"/>
   <rdf:range rdf:resource="#Activity"/>
   <dependOn rdf:resource="locationIn"/>
   <dependOn rdf:resource="lightLevel"/>
    . . .
</owl:ObjectProperty>
```

위의 OWL 표현은 사용자의 현재 활동 상태가 Watching-TV인 actStatus 프로퍼티는 Person과 TV의 위치, TV의 상태, 방의 밝기와 의존관계가 있음을 나타낸 것이다.

## 4. 미들웨어 설계 및 구현

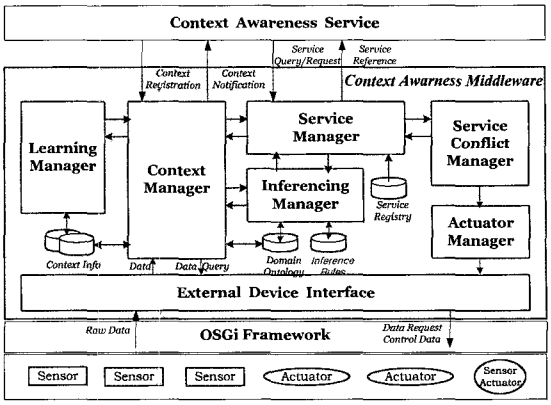
### 4.1 미들웨어 설계

본 절에서는 온톨로지 기반 상황 인식 모델을 기반으로 미들웨어를 설계 및 구현한다. 상황 인식 미들웨어는 OSGi 프레임워크 기반의 번들로 설계하였으며 BundleActivator 인터페이스를 상속받고, 온톨로지 추론을 위해 Jena.jar 파일을 이용한다. 설계한 미들웨어는 센서로부터 받은 데이터를 이용해서 상황 정보를 관리, 조합, 학습, 추론하여 이 정보를 원하는 상황 인식 서비스에 전달하는 역할을 담당한다.

설계한 미들웨어는 상황 관리자(context manager), 서비스 관리자(service manager), 추론 관리자(inferencing manager), 서비스 충돌 관리자(service conflict manager), 학습 관리자(learning manager), 엑츄레이터 관리자(actuator manager), 외부장치 인터페이스(external device interface)로 구성되며 전체 구성은 (그림 3)과 같다.

#### 4.1.1 상황 관리자

상황 관리자는 기본 상황을 생성하는 상황 생성기(context generator)와 기본 상황을 저장, 관리하고 복합 상황 추론을 위해 추론 관리자와 상호 작용을 하는 온톨로지 관리 모듈(ontology management module), 그리고 상황 인식 서비스와 공급/등록 방식으로 이벤트 전달을 하는 이벤트 브로커(event broker), 그리고 상황 저작자와 상황 인식 서비스로부터 전달받은 명령을 해석하기 위한 번역기(translator)로 구성되며 저작자는 계층적 상황 온톨로지를 이용해서 도메인에 필요한 상황 온톨로지를 OWL 파일로 정의하고 이를 번역기에 전달한다.




(그림 3) 상황 인식 미들웨어 구성도

#### 4.1.2 서비스 관리자

서비스 관리자는 미들웨어에 연결된 상황 인식 서비스들 간의 상호 작용을 지원하기 위해서 서비스 검색 및 등록, 조합, 삭제 기능을 제공하며 다중 사용자의 서비스 요구를 처리한다. 사용자로부터 서비스에 대한 요청이 들어오면 서비스의 종류와 사용자 프로파일, 환경의 상황 정보에 따라 서비스를 제공하기 위한 계획을 세우고 서비스 객체를 생성한 후 SubTask들로 Task를 구성한다. Task가 서비스에 등록되면 서비스 충돌 관리자를 통해 충돌 여부를 확인하고 Task를 실제 엑츄레이터의 동작을 관리하는 엑츄레이터 관리자에 등록한다.

서비스는 Task의 묶음으로 한 명의 사용자가 하나 이상의 서비스 객체를 가지며, Task는 일정한 목표를 가지고 있는 서비스 단위로 하나 이상의 Task 또는 가장 작은 서비스 단위인 SubTask로 구성된다. 예를 들어 'TV 시청'이라는 서비스는 'TV power On', 'lightLevel Dim'라는 Task와 'TV channel 11', 'TV soundLevel 15'라는 SubTask로 구성된다.

#### 4.1.3 추론 관리자

기존 추론엔진이 단순한 상황 브로커나 온톨로지에 기반한 상황 쿼리 엔진(context query engine) 형태인 것에 비해 추론 관리자는 온톨로지 추론을 위해서 Jena API를 이용하여 OWL 상황 온톨로지 파일을 파싱하고 Fact인 nTriple형태로 변환한 후 뒤 룰 기반의 추론 시스템인 JESS(Java Expert System Shell)로 복합 상황 추론을 한다. 이때, 기본 상황 생성기에서 발생한 기본 상황을 Fact의 형태로 추론엔진에 추가 및 삭제하며, 상황 저작자가 정의한 규칙들과 온톨로지를 이용해서 온톨로지 추론과 사용자 규칙 기반 추론을 통해서 상황 정보의 일관성 검사와 상황 관계성을 도출하고 룰에 따른 최적 서비스 검출 및 서비스 충돌 검사를 보조한다.

4.1.4 서비스 충돌 관리자

사용자에 대한 서비스 제공 시에 발생하는 Task, subTask의 충돌을 관리하며 사용자 등급, 공간별 소유자(owner), 공간별 사용자 우선순위(priority), 서비스/타스크 우선순위, 디바이스 우선순위를 조합한 우선순위 기반의 서비스 룰을 구현하여 충돌을 해결한다.

서비스 충돌은 한 공간에서 다중 사용자가 같거나 유사한 Task들을 동시에 제공받고자 할 때 발생하는 다중 사용자 간의 서비스 충돌과 여러 Task가 한 사용자에 의해 동시에 발생할 때 일어나는 Task 충돌 (취침서비스 + TV보기 서비스), 동시에 하나의 기기를 서로 다른 서비스가 제어하려 하거나 또는 비슷한 환경을 제어(온도를 올리는 방법 : 보일러 On, 에어컨 Off, 창 Open/Close)하려고 하는 기기 사이의 충돌인 SubTask 충돌로 분류된다.

4.2 상황 추론

설계한 상황 인식 모델은 도메인 내의 상황 정보를 추론하는 방법으로 온톨로지 추론(ontology reasoning), 규칙기반 추론(rule-based reasoning), 확률기반 추론(probabilistic reasoning)을 지원한다.

4.2.1 온톨로지 추론

상황에 대한 추론은 1차 술어 논리 기반의 규칙 기반 추론을 하며 온톨로지 리스너(reasoner)와 규칙 기반 리스너는 Jena API를 기반으로 구현했다. 온톨로지 추론은 도메인 온톨로지가 전환되거나 통합될 때 내부 상황 클래스의 일관성과 그들 간의 함축된 관계 등을 찾아내는데 사용된다.

온톨로지 추론은 RDFS 추론과 OWL 추론을 포함하며 다음은 온톨로지 추론 규칙을 나타낸 것이다.

〈표 3〉 온톨로지 추론 규칙 예

| 구분 | 온톨로지 추론 규칙 |
|---|---|
| subClass Of | (?A rdfs:subClassOf ?B), (?B ?rdfs:subClassOf ?C) ⇒ (?A rdfs:subClassOf ?C) |
| subPropert yOf | (?A rdfs:subPropertyOf ?B) ∧ (?B rdfs:subPropertyOf ?C) ⇒ (?A rdfs:subPropertyOf ?C) |
| Transitive Property | (?P rdf:type owl:TransitiveProperty),(?A ?P ?B),(?B ?P ?C) ⇒ (?A ?P ?C) |
| inverseOf | (?A owl:inverseOf ?B) ∧ (?A ?X ?Y) ⇒ (?Y ?B ?X) |

아래의 OWL 표현은 'Jung'이라는 사용자가 BedRoom에 있다는 상황 정보가 발생하면 온톨로지 TransiviePropert 규칙을 이용한 상황 추론을 통해서 BedRoom이 indoorSpace의 부분임을 유추함으로써 'Jung'이 집안에 있다는 것을 알 수 있다.

```
<owl:ObjectProperty rdf:ID="locatedIn">
<rdf:type rdf:resource="&owl;TransitiveProperty"/>
<rdfs:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<inha:Room rdf:ID="BedRoom">
   <inha:locateIn rdf:resource="#indoorSpace"/>
</inha:Room>
```

4.2.2 규칙 기반 추론

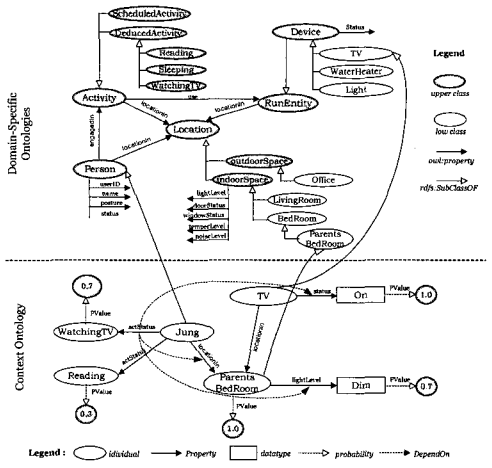
사용자 정의 규칙 기반 추론은 순방향 체이닝(forward chaining)방법을 사용하며 다음과 같은 사용자 정의 규칙 셋(rule-set)에 의하여 사용자의 상태를 추론 할 수 있다.

〈표 4〉 사용자 정의 추론 규칙

| Context | Inference Rules |
|---|---|
| Sleeping | (?p locatedIn BedRoom), (BedRoom lightLevel Dark), (BedRoom noiseLevel Low) ⇒ (?p actStatus Sleeping) |
| WatchingTV | (?p locatedIn BedRoom), (TV status On) ⇒ (?p actStatus WatchingTV) |
| Showering | (?p locatedIn BathRoom), (BathRoom lightLevel Bright), (BathRoom noiseLevel High), (WaterHeater status On), (Bathroom doorStatusl Closed) ⇒ (?p actStatus Showring) |
| Washing | (?p locatedIn BathRoom), (BathRoom lightLevel Bright), (BathRoom noiseLevel Medium) ⇒ (?p actStatus Washing) |

4.2.3 확률 기반 추론

확률 기반 추론은 확률 확장 마크업 요소인 dependOn속성에 의해 서로 의존성이 있는 다른 클래스의 인스턴스의 현재 상황 정보를 파악하여 추론하는 과정을 (그림 5)에서 보여주고 있다. 상황 온톨로지를 통해 사용자의 위치 및 TV의 상황 정보의 수가 1비트 이므로 0 아니면 1 또는 On 아니면 Off의 값을 갖지만, BedRoom의 조명 상태는 2비트 (Dark, Dim, Bright, veryBright)로 정보를 나타낸다. 위와 같이 영향을 미치는 프로퍼티의 값으로 확률 계산을 해서 현재 사용자의 상태를 추론한다.



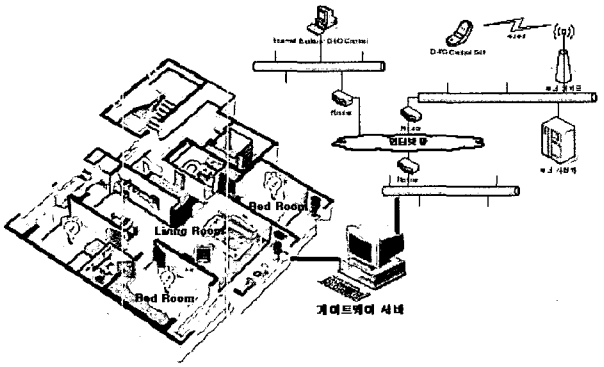

(그림 4) 확률 추론을 위한 온톨로지

확률 추론을 위한 도메인 온톨로지는 (그림 4)와 같이 Person은 'Jung'이며, Location은 indoorSpace의 내부인 BedRoom을 상속한 ParentsBedRoom에 있고 TV는 On 상태이며, ParentsBedRoom의 밝기는 어두운 상태이다. 현재의 상황을 가지고 확률을 계산함으로써 사용자의 활동 상태가 Reading이 아닌 WatchingTV라는 것을 추론 할 수 있다. 또한, 상황을 온톨로지로 모델링함으로써 BedRoom를 상속해서 동일한 구조을 가지는 ParentsRoom이나 ChildRoom등과 같은 여러 클래스를 쉽게 구성할 수 있는 재사용성을 보장해 준다.

## 5. 실험 및 평가

이 장에서는 이 논문에서 설계한 상황 인식 미들웨어의 기능과 성능을 평가하기 위해서 원격지에서 휴대폰 및 웹을 통해 스마트 홈 내의 상황을 인식하고 적절한 서비스를 실행하는 실험을 진행하였다, 홈 네트워크 서비스 환경에 사용된 게이드웨이 서버는 Intel P4 2.4GHz, 메모리 1GB, 윈도우즈 서버 2003 환경에서 OSGi 프레임워크인 Knopflerfish 1.3.3을 사용하고, HP의 시맨틱 웹 툴킷인 Jena2를 이용하였다. 게이트웨이 서버와 스마트 홈과의 인터페이스를 제공하는 Digital I/O Controller는 Redhat Linux 9.0, GCC Compiler와 SNDS50100 ARM7TDMI Core (50Mhz), 2MBytes Flash를 사용하였으며 휴대폰으로 스마트 홈을 제어하는 소프트웨어는 JAVA SDK 1.4.2_12, WIPI XCE SDK 2.0.2를 사용하여 구현하였다.

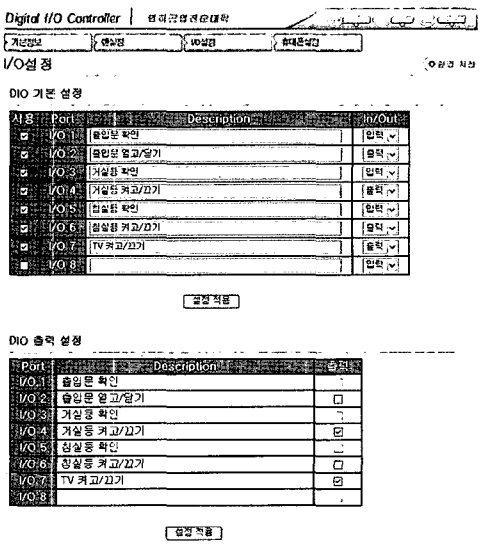
실험 시나리오는 사용자가 원격지에서 웹이나 휴대폰으로 홈 내의 상황 정보를 확인하고 이에 따라서 사용자의 의도가 집안에 있는 상황 인식 서비스에 적절하게 전달되는지를 평가한다. 실제 가정환경에서 제공될 수 있는 서비스는 그 종류가 많기 때문에 본 논문에서는 원격 상황 인식 제어 서비스를 아래와 같이 출입문, 거실 전등, 침실 전등, TV 제어로 한정하여 가정환경을 구성하였다. 구현된 시스템의 원격 상황 인식 제어 서비스의 종류는 불 켜기/끄기, 문 열기/닫기로 최대 8개까지 가능하다.



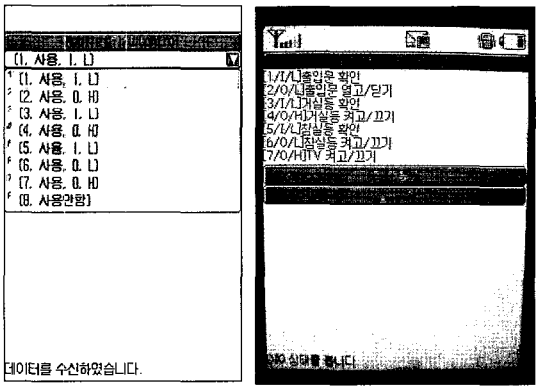

(그림 5) 원격 상황 인식 제어 서비스 실험 환경

(그림 6)은 웹을 통하여 접속한 화면을 보여주고 있으며 DIO(Digital Input Output) 기본 설정 화면의 8개 I/O포트 중 2, 4 ,6, 7번이 출력으로 설정되어 있으며 DIO 출력 설정은 아직 설정 적용 버튼을 선택하지 않았기 때문에 4, 7번 포트만 출력으로 설정되어 있는 것을 확인할 수 있다.

(그림 7)은 외출한 가족이 열쇠가 없어서 집에 들어오지 못하는 상황을 가정하여 출입문과 거실 전등, 거실 TV를 켜기로 설정한 후, 스마트 홈의 환경 디바이스의 상황을 수




(그림 6) 웹에서의 DIO 설정 화면




(그림 7) 휴대폰에서의 상황 인식 서비스 제어

〈표 5〉 I/O포트별 디바이스 대응표

| Port | Device | Description |
|---|---|---|
| I/O 1 | Door status | |
| I/O 2 | Door Open/Close | H : Door Open |
| I/O 3 | LivingRoom Light status | |
| I/O 4 | LivingRoom Light On Off | H : Light ON |
| I/O 5 | BedRoom Light status | |
| I/O 6 | BedRoom Light On Off | |
| I/O 7 | TV On Off | H : TV ON |

신하여 보여주고 있으며 각 포트별로 대응된 홈 내의 디바이스는 <표 5>와 같으며 응용에 따라서 포트에 대응된 환경 디바이스를 동적으로 필요에 따라 등록하여 제어할 수 있다.

## 6. 결 론

이 논문에서는 유비쿼터스 컴퓨팅 환경에서 온톨로지 기반 상황 인식 서비스 관리 모델을 제시하였고, 제시한 모델을 이용해서 상황 인식 서비스 미들웨어를 설계 및 구현하였다. 설계한 미들웨어는 상황 인식 서비스가 동적으로 필요로 하는 상황 온톨로지를 미들웨어에 등록할 수 있도록 함으로써 상황 인식 서비스별로 개별화된 상황 정의를 지원하며 설계된 미들웨어는 센서와 가전 등의 물리적 기기와 연결하기 위해서 OSGi 프레임워크 환경에서 구현하였다. .

이 논문에서 설계/구현한 미들웨어의 타당성을 입증하기 위해서 스마트 홈의 상황을 원격에서 감시 및 제어 할 수 있도록 상황 인식 서비스를 구현하고 실험을 통해서 미들웨어의 기능과 성능을 평가하였다. 이 논문에서의 OSGi 기반 상황 인식 미들웨어는 홈네트워크, 텔레매틱스, 스마트 오피스 등의 다양한 유비쿼터스 환경에서 서비스 게이트웨이에 탑재되어서 상황 인식 서비스 운용 환경에 적용될 수 있을 것으로 보인다.

향후 연구 방향은 정확하지 않은 센싱 데이터로 인해 발생하는 불확실(imperfect)하고 모호한(uncertain) 상황에 대한 추론 및 처리에 대한 연구이다.

## 참 고 문 헌


[1] M. Weiser and J. S. Brown, The Coming Age of Calm Technology, In P. J. Denning & R. M. Metcalfe(Eds.), Beyond Calculation: The Next Fifty Years of Computing, pp.75-85, 1998.

[2] A. D. Norman, The Invisible Computer: Why Good Products Can Fail, The Personal Computer Is So Complex, and Information Appliances Are the Solution, MIT Press, 1998.

[3] J. Wejchert, "The Disappearing Computer,"Information Document, IST Call for proposals, European Commission, Future and Emerging Technologies, 2000.

[4] 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

[5] B. N. Schilit, N. Adams, and R. Want, "Context-aware computing applications," Proceedings of the Workshop on Mobile Computing System and Applications, pp. 85-90, 1994.

[6] A. K. Dey, "Supporting the Construction of Context-Aware Applications," Dagstuhl seminar on Ubiquitous Computing, 2001.

[7] Chen, Harry, Tim Finin, and Anupam Joshi. "An Intelligent Broker for Context-Aware Systems." Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA, October 12-15, 2003.

[8] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. "Gaia: A Middleware Infrastructure to Enable Active Spaces", In IEEE Pervasive Computing, pp.74-83, Oct-Dec., 2002.

[9] Tao Gu; Hung Keng Pung ; Da Qing Zhang , "Toward an OSGi-Based Infrastructure for Context-Aware Applications", Pervasive Computing, IEEE, volume 3, issue 4, 66-74 pages, Oct., 2004.

[10] Tao Gu, Hung KP, Da QZ, "A Service-oriented. middleware for building context-aware services", Journal. of Network and Computer Applications, Vol.28, 2005

[11] OWL : http://www.w3.org/2004/OWL/

[12] 이승근, "유비쿼터스 컴퓨팅 환경을 위한 온톨로지 기반 상황 인식 서비스 미들웨어", 인하대학교 박사학위 논문, 2006.

[13] S. Shafer, B. Brumitt, and. B. Meyers, "The EasyLiving Intelligent Environment System". CHI Workshop on Research Directions in Situated Computing, April, 2000.

[14] MicroSotf EasyLiving http://research.microsoft.com/easyliving.

[15] K. Cory, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, and W. Newstetter, "The aware home: A living laboratory for ubiquitous computing research", In the Proceedings of the Second International Workshop on Cooperative Buildings, 1999.

[16] http://www-static.cc.gatech.edu/fce/ahri

[17] Yatani, Koji, Masanori Sugimoto and Fusako Kusunoki. "Musex: A System for Supporting Children's Collaborative Learning with PDA's." Second IEEE International Workshop on Wireless and Mobile Technologies in Education. 2004.

[18] L. Gong, "A Software Architecture for Open Service Gateways," IEEE Internet Computing, Vol.5, No.1, pp. 64-70, 2001.

[19] P. Dobrev, D. Famolari, C. Kurzke, and B. A. Miller, "Device and Service Discovery in Home Networks with OSGi," IEEE Communications Magazine, Vol.40, No.8, pp. 86-92, 2002.


## 정 헌 만

e-mail : hmjung@inhatc.ac.kr

1996년 서울산업대학교 전자계산공학과 (학사)

2001년 인하대학교 대학원 전자계산공학과(공학석사)

2004년 인하대학교 대학원 전자계산공 학과(박사수료)

2001년~현재 경인여대, 인하전문대 강사

관심분야 : 상황인식, 유비쿼터스, 센서네트워크, 시맨틱웹, 웹서비스



## 이 정 현

e-mail : jhlee@inha.ac.kr

1977년 인하대학교 전자공학과(학사)

1980년 인하대학교 대학원 전자공학과 (공학석사)

1988년 인하대학교 대학원 전자공학과 (공학박사)

1979년~1981년 한국전자기술연구소 시스템 연구원

1984년~1989년 경기대학교 전자계산학과 교수

1989년~현재 인하대학교 컴퓨터공학부 교수

관심분야 : 자연어처리, HCI, 정보검색, 유비쿼터스 컴퓨팅