韓國 컴퓨터情報學會 論文誌
第11卷 第4號, 2006. 9.

2006-11-4-1-19

상황인식 서비스 응용을 위한 동적 서비스 관리 모델 설계

정 헌 만*, 이 정 현**

Design of Dynamic Service Management Model for Context-Aware Service Applications

Heon-Man Jung*, Jung-Hyun Lee**

요 약

상황 인식 서비스는 상황 획득 및 추론 과정 등의 기능을 지원함으로써, 상황 인식 서비스의 구현을 쉽게 하는 연구들이 진행되어 왔다. 그러나, 이들 연구에서는 유비쿼터스 컴퓨팅 환경에서 필수적으로 지원되어야 하는 사용자 및 서비스의 이동성을 효과적으로 지원하지 못하고 있다. 따라서, 본 논문에서는 서비스의 이동성을 효과적으로 지원하기 위해서 서비스간의 상호 작용을 위한 서비스 검색 및 조합, 서비스 이동성을 지원하는 동적 상황 인식 서비스 모델을 제안하고 이 모델을 기반으로 상황 인식 미들웨어를 설계 구현한다. 또한 설계하는 미들웨어는 OSGi 프레임워크 위에서 구현함으로써 UPnP, Jini 등의 표준 인터페이스 기술을 이용해서 다양한 가전과 센서 등의 외부 기기들과 연동이 가능하다.

Abstract

As context aware service supports a process of context acquisition and reasoning, there are many studies to facilitate the implementation of context aware service. However, these studies have not supported efficiently a user or service mobility that should be supported necessarily in ubiquitous computing environment. Therefore, this study proposes a dynamic context aware service model which supports a dynamic management of context information, service retrieval and composition for interactions between services, and service mobility. Then we design a middleware based on this model and implement the middleware. As the middleware is implemented on the OSGi framework, it can have an interoperability between the devices such as computers, PDA, home appliances, and sensors, because of using the standard interface technologies like UPnP, Jini and so on.

▸ Keyword : 상황인식 모델(Context-Aware Model), 상황 인식 미들웨어(Context-Aware Middleware), OSGi(Open Services Gateway Initiative), 유비쿼터스 컴퓨팅(Ubiquitous Computing)

• 제1저자 : 정헌만
• 접수일 : 2006.08.14, 심사일 : 2006.09.01, 심사완료일 : 2006.09.20
* 경인여대,인하전문대 강사 ** 인하대학교 컴퓨터공학부 교수

Ⅰ. 서 론

유비쿼터스 컴퓨팅은 물리적인 공간을 능동적이고 지능적인 환경, 즉 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, 해석, 추론을 거쳐 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].

이러한 유비쿼터스 컴퓨팅 시스템 환경에 관련한 많은 연구들이 컴퓨팅 환경, 서비스 플랫폼, 프레임워크, 가전 환경등으로 진행되고 있으며 디바이스의 이종성 및 이동성 지원, ,서비스의 분산성 및 이동성 지원, 사용자 이동성 지원, 외부 환경의 장비들과의 손쉬운 상호 연동성이 유비쿼터스 컴퓨팅 환경에서 제공해야 할 기능적 요구사항으로 제시 되었다.[6].

그러나, 기존 연구들에서는 이들 요구사항들을 모두 만족시키지 못하고 있다. 예를 들어서, 프레임워크 연구에서는 적응성 지원이나 상황 인식 지원, 응용 개발 환경을 지원하나 사용자 이동성에 관한 지원이 부족하며, 서비스 플랫폼에 관한 연구에서는 디바이스 이동성과 사용자 상황 관리 등에 관한 부분이 부족하다[7,8,9].

따라서, 본 논문에서는 앞에서 기술한 유비쿼터스 컴퓨팅 환경의 요구사항들을 효과적으로 지원할 수 있는 동적 상황 인식 서비스 관리 모델을 제안하고, 이를 지원하기 위해서 상황 관리자와, 서비스들간의 상호 작용을 지원하기 위한 서비스 관리자, 그리고 사용자와 디바이스의 이동성을 효과적으로 지원하기 위한 서비스 이동 관리자로 구성된 상황 인식 서비스를 위한 미들웨어를 설계 및 구현한다.

제안하는 동적 서비스 관리 모델은 미들웨어내에 있는 서비스들을 기존 구문 검색 방식이나 온톨로지 기반 검색의 정적인 서비스 기술 내용뿐 만 아니라 현재 서비스의 상황 정보를 검색에 이용함으로써 검색 결과를 동적으로 얻을 수 있게 함으로써 보다 정확한 서비스 검색을 가능하게 하며, 이들 서비스를 동적으로 조합해서 새로운 가상 서비스를 생성하고 이를 제공함으로써 보다 유연한 서비스간 상호 작용을 지원하도록 한다.

또한, 전체 미들웨어를 OSGi 프레임워크 국제 표준 기술 위에서 설계 및 구현함으로써 다양한 외부 기기들과 손쉽게 연동을 지원하고, 이를 통해서 미들웨어상의 여러 상황 인식 응용들이 사용자에게 다양한 기기들을 이용한 서비스를 효과적으로 제공할 수 있게 한다.

Ⅱ. 관련 연구

이 장에서는 유비쿼터스 컴퓨팅에서 필수적으로 요구되는 상황 인식 관련 기술과 모델링 방법, 상황 인식 관련 기존 연구에 대해서 기술한다.

2.1 상황 인식

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[10].

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[4,11].

2.2 상황 인식 시스템

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

Gaia[12]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)[13,14]은 유비쿼터스 환경에서 제공 되어지는 다양한 상황의 상호의존적 개념에 대한 정의 및 모바일 환경에서의 상황인식 서비스 제공을 위해 구현되었으며 OWL[15]기반의 온톨로지를 사용하여 의미기반 상황 표현과 다양한 형태의 상황에 대한 추론, 지식 공유, 상황의 분류와 상호 의존성과 관련된 문제를 다루고 있다.

사용자 신원, 위치, 대상물 인식 정보를 이용하여 정보가전기기를 제어하는 지능형 가정환경을 구축을 목표로 하는 MicroSoft의 Easy Living[16,17]은 객체와 사람, 공간사이의 물리적 관계의 정보를 상황정보화 하여 사용함으로써 상황을 추론하고 사용자가 원하는 서비스를 제공한다. 혼자 사는 노인들의 위치 정보와 활동 상태 정보를 파악하여 위급 상황 발생 시에 병원이나 보호자에게 알려주는 스마트 홈 모델을 제시한 Georgia Tech. Aware Home Research Initiative(AHRI)의 AwareHome[18,19]는 가정 내 사용자의 상황(누가, 어디에서, 무엇)을 파악하기 위한 실내에서의 위치 인식을 위해 RFID와 마루 메트를 이용했으며, Musex[20]는 박물관에서 관람하고 있는 어린이를 대상으로 실시한, 학습자 위치 인식에 맞춰 학습자 부근의 전시물과 관련된 컨텐츠를 RFID가 부착된 PDA(Personal Digital Assistants)에 퀴즈 형태로 제공하여 사용자의 흥미를 유발시키도록 하는 연구이다.

현재 상황 인식 서비스에 대한 다양한 연구가 진행되고 있으나 사용자 및 서비스의 이동성 지원 및 정확한 서비스의 발견에 대한 연구는 부족한 실정이다. 따라서 본 논문에서는 서비스의 이동성을 효과적으로 지원하기 위해서 서비스간의 상호 작용을 위한 서비스 검색 및 조합, 서비스 이동성을 지원하는 동적 상황 인식 서비스 모델을 제안하고 이 모델을 기반으로 상황 인식 미들웨어를 설계 구현한다.

III. 동적 서비스 관리 모델

이 장에서는 유비쿼터스 컴퓨팅 환경에서 상황 인식 미들웨어 설계에 필요한 동적 서비스 관리 모델을 제안한다. 제안하는 동적 상황 인식 서비스 모델은 사용자와 디바이스 등의 이동에 따른 상황 인식 응용의 동적 이동을 효과적으로 지원하는데 목적이 있다 또한 온톨로지를 기반으로 상황을 모델링함으로써 효과적인 추론, 사람과 기계, 기계사이의 원활한 커뮤니케이션, 확장성 및 상황정보 공유의 용이성, 다양한 응용 서비스 관리의 편리함을 얻을 수 있다.

3.1 상황 온톨로지 구성

그림 1은 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.

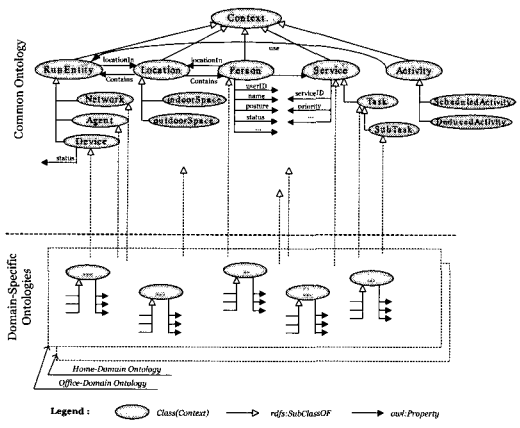


그림 1 . 계층 상황 온톨로지 구성

Fig 1 . Hierarchical Context Ontology Structure

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.

3.2 상황 정보의 구성

상황 인식 시스템에서 사용되는 상황 정보는 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 실세계와 유사한 상황을 유지하기 위해 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 1 사이의 실수 값으로 정의 하며, 계산 양이 적으면서 비교적 성능이 좋은 사다리꼴 모양의 퍼지 소속도 함수를 사용한다.

상황 정보는 연속적인 데이터와 이산적인데이터로 구분하여 데이터 타입에 따라 적절한 전처리과정이 진행될 수 있도록 한다.

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 동적 서비스 관리 모델

이 논문에서 제안하는 상황정보 기반 서비스 검색 및 조합을 위해서 필요한 온톨로지를 정의한다. 서비스는 서비스 그라운딩과 서비스 QoS, 서비스 타입 세 개의 요소를 갖는다. 서비스 그라운딩은 입력과 출력으로 구성되며 각각 이름, 역할, 단위, 타입 속성을 갖는 파라미터로 구성되며 서비스 QoS는 서비스 실행 시간, 신뢰도, 실행 비용으로 구성된다. 마지막으로 서비스 타입은 도메인별로 정의될 수 있다. 그림 2는 전체 서비스 온톨로지의 개요를 설명하고 있다.

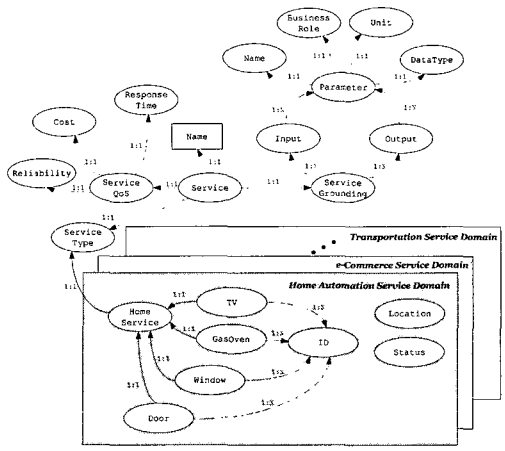


그림 2. 서비스 기술을 위한 온톨로지

Fig 2. Ontology for Service description

그림 2는 Home Automation 서비스 도메인을 예로 구성한 것이다. 예에서 홈 자동화 서비스는 TV, 가스오븐, 창문, 출입문 서비스가 있으며 위치, 상태 등의 상황 정보를 정의한다. 정의된 온톨로지는 다음에 설명하는 서비스 발견 및 조합을 위해서 사용된다.

3.3.1 서비스 발견

서비스 발견은 서비스 요청자의 질의에 따라서 적절한 서비스를 리턴해 주는 역할로써, 이를 위해서는 서비스 제공자는 서비스 레지스트리에 그림 3의 온톨로지를 이용해서 서비스를 등록한다. 이 논문에서 설계하는 서비스 관리자는 서비스 요청자의 질의를 분석해서 서비스 타입과 서비스 그라운딩 요소의 매치를 통해서 등록되어 있는 서비스 중 가장 적합한 서비스를 발견 한 후 리턴한다. 만약, 적합한 서비스가 발견되지 않으면 서비스 조합을 실행한다. 그림 3은 서비스 발견 과정을 대략적으로 표현한 것이다.

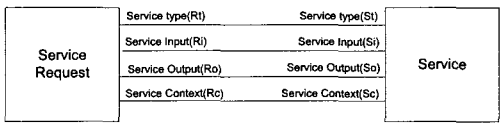


그림 3. 서비스 매칭 방법

Fig 3. Service macthing method

서비스 매칭 방법은 서비스 타입, 입력, 출력, 그리고 상황 정보를 이용하며 서비스 타입 온톨로지 모델은 서비스가 응용에서 가질 수 있는 서비스 타입을 나타낸다. 서비스 도메인마다 다를 수 있으며, 그림 2에서 "Home Service"에 4가지 서비스로 표현하였다. 서비스의 입력과 출력은 서비스 그라운드에 속한 온톨로지로 서비스는 호출되기 위해서 WSDL 인터페이스로 매핑된다. 실제적인 호출은 인터페이스에 있는 연산의 호출을 위해서 서비스의 연산에 전달하는 파라미터에 의해서 이루어진다. 서비스의 입력과 출력은 WSDL의 매시지로 매핑되는데 이때, 입력과 출력은 파라미터를 포함하며, 파라미터는 이름, 역할, 단위, 데이터 타입을 갖는다. 서비스 상황은 사용자가 원하는 서비스를 찾기 위해서 정의될 수 있다. 이때, 가전기기나 제어대상의 위치, 상태 등이 포함된다. 그림 4는 서비스 온톨로지와 웹서비스에 사용되는 WSDL(Web Services Description Language)[21]과의 매핑을 설명한 것이다.

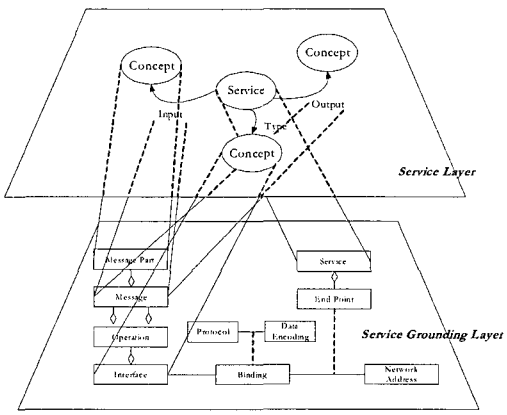


그림 4. 서비스 모델과 서비스 그라운딩과의 관계
Fig 4. Relation to Service Model and Service Ground

서비스 매칭은 서비스 요청과 등록된 서비스 기술의 온톨로지 매핑에 의해서 이루어진다. 이 연구에서 제안하는 매칭 알고리즘은 서비스 요청자가 원하는 서비스 타입, 서비스 입력과 출력, 상황 정보를 서비스 제공자의 기술과 순서대로 매칭함으로써 정확한 서비스 제공자를 찾아낸다.

3.3.2 서비스 조합

사용자가 원하는 서비스가 검색되지 않으면 서비스들을 조합해서 원하는 서비스를 제공해 줄 수 있다. 이때, 시스템은 두 서비스가 조합이 가능한지 여부를 판단 할 수 있어야 한다. 두 개의 서비스가 조합 가능한지를 알기 위해서 먼저 하나의 서비스가 다른 서비스의 연산을 호출 할 수 있는지 판단한다. 연산의 호출은 메시지 전달에 의해서 이루어지며, 이를 위해서 서비스를 구성하는 연산과 메시지의 상호운영성을 판단해야 한다.

조합된 서비스들은 트랜잭션의 시간이 길 수 있다. 따라서, 가장 최적화된 실행 시간을 갖는 조합 계획을 갖는 것은 매우 중요하다. 서비스의 조합 계획이 복수개가 발견되는 경우, 각각의 서비스 조합 계획 중 최적의 계획을 선택해야 한다. 이를 위해서 적절한 QoS 측정 방법이 필요하다. 이 연구에서는 서비스의 신뢰성, 실행 시간과 실행 비용의 측면으로 평가하며 평가 방법은 다음과 같다. 식 1은 전체 서비스 조합의 QoS값으로 이 값을 비교해서 최적의 서비스 조합을 찾는다. QtValue값은 서비스의 신뢰성에 비례하고 실행 시간과 비용에 반비례한다.

$$QtValue(S) = \frac{\dim(S, \mathrm{Re}liability)}{\dim(S, Time) * \dim(S, Cost)}$$

$$\dim(S, \dim) = \sqrt[3]{Min(S, \dim) * Avr(S, \dim) * Max(S, \dim)}$$

..... (식1)

서비스의 신뢰성은 전체 실행 횟수에서 서비스의 실패한 횟수의 비율로 표현될 수 있으며, 전체 조합 계획의 신뢰성은 조합에 참여하는 서비스들의 신뢰성의 곱으로 표현되며, 실행 시간과 비용은 각 서비스의 실행 시간의 합으로 식 2로 계산된다.

$$\dim(CompositeS, \mathrm{Re}liability) = \prod_{i=1} \dim(S_i, \mathrm{Re}liability)$$

$$\dim(CompositeS, Time) = \sum_{i=1} \dim(S_i Time)$$

$$\dim(CompositeS, Cost) = \sum_{i=1} \dim(S_i Cost)$$

..... (식2)

IV. 설계 및 구현

본 절에서는 동적 서비스 관리 모델을 기반으로 동적 유비쿼터스 공간을 구성하기 위한 상황 인식 미들웨어를 설계

및 구현한다. 설계한 미들웨어는 다양한 프로토콜을 이용해서 외부 센서 및 가전등과 연결을 지원하는 외부 장치 인터페이스와 기능과 센서로부터 획득된 데이터를 통해서 상황을 유추하고 관리할 수 있는 상황 관리 기능, 미들웨어를 사용하는 상황 인식 응용들이 서비스를 효과적으로 검색하고 필요한 서비스들을 조합할 수 있는 서비스 관리 기능, 그리고, 사용자의 이동에 따라서 미들웨어간에 서비스들이 동적으로 재구성될 수 있는 서비스 이동 기능으로 구성한다.

4.1 동적 서비스 지원 미들웨어 설계

설계한 미들웨어의 전체 구성은 그림 5와 같다. 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.

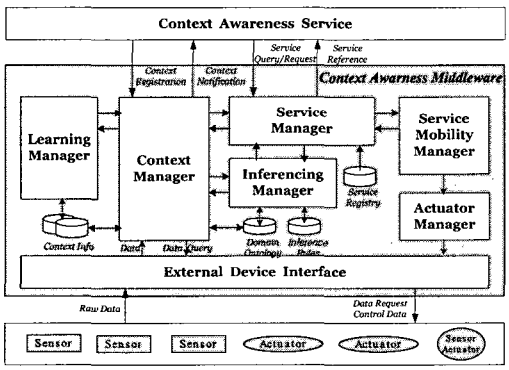


그림 5. 동적 서비스 지원 미들웨어 구성도

Fig 5. Middleware Architecture support for Dynamic Services

상황 관리자(context manager)는 기본 상황을 생성, 저장, 관리하고 복합 상황 추론을 위해 추론 관리자와 상호작용을 하며 상황 인식 서비스와 공급/등록 방식으로 이벤트 전달을 담당한다. 그리고 상황 저작자와 상황 인식 서비스로부터 전달받은 명령을 해석한다.

추론 관리자(inferencing manager)는 추론엔진이 단순한 상황 브로커나 온톨로지에 기반한 상황 쿼리 엔진(context query engine) 형태인 것에 비해 온톨로지 추론을 위해서 Jena API를 이용하여 OWL 상황 온톨로지 파일을 파싱하고 Fact인 nTriple형태로 변환한 후 뒤 룰 기반의 추론을 한다.

학습 관리자(learning manager)는 동적으로 변화하는 환경에서 사용자의 행동 패턴 및 프로파일을 기반으로 사용자에게 적절한 서비스를 제공하기 위해서 신경망(neural network), 결정트리(decision tree), 연합 룰 찾기(association rule finding) 등을 적용 시켜서 기계학습을 수행한다.

이 논문에서는 전체 미들웨어 중 서비스 관리자와 서비스 이동 관리자를 중심으로 설계한다.

4.2 서비스 관리자

서비스 관리자(service manager)는 미들웨어에 연결된 상황 인식 서비스들 간의 상호 작용을 지원하기 위해서 서비스 검색 및 등록, 조합, 삭제 기능을 제공하며 다중 사용자의 서비스 요구를 처리한다. 시스템 관리자는 온톨로지 기반 검색을 확장해서 상황 정보를 고려한 온톨로지 기반 서비스 검색 기능을 제공하며, 기본 서비스들의 조합을 통해서 동적으로 서비스 조합 계획을 생성하고 관리할 수 있도록 설계한다. 그림 6은 전체 서비스 관리자 구성도이다.

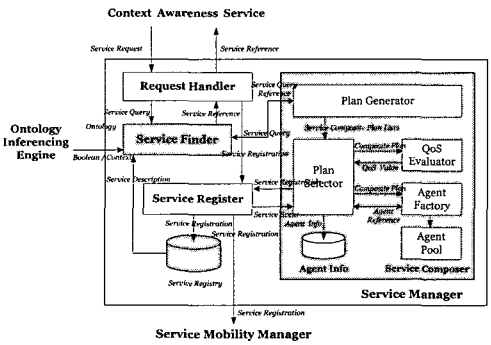


그림 6. 서비스 관리자 구성도

Fig 6. Architecture of Service Manager

서비스 관리자는 서비스 검색 및 서비스 조합 기능을 수행하며 요구 처리기, 서비스 비교기, 서비스 등록기, 서비스 조합기, 서비스 레지스트리로 구성된다. 요구 처리기는 상황 인식 서비스와의 인터페이스를 담당하는 역할을 수행한다. 전달되는 요청의 종류에 따라서 서비스 요청사항을 서비스 비교기와 등록기에 전달하며 상황 인식 서비스는 서비스를 서비스 등록기를 이용해서 서비스 레지스트리에 저장한다. 상황 인식 서비스가 미들웨어에 위치한 다른 상황 인식 서비스의 검색을 요구는 서비스 비교기에 전달되며, 서비스 레지스트리에 등록된 정보와 서비스 질의 온톨로지를 추론

엔진을 이용해서 비교하고, 일치하는 서비스의 레퍼런스를 전달한다. 서비스 조합기는 서비스 레지스트리에 요청한 서비스가 존재하지 않는 경우에는 레지스트리에 등록된 서비스들을 조합해서 요청한 서비스와 동일한 결과를 얻을 수 있도록 서비스 조합 계획과 실행을 관리하는 역할을 수행한다. 서비스 조합기에 의해서 생성된 가상 서비스는 서비스 요청자에게 전달되어서 다른 서비스와 동일한 방식으로 호출되도록 한다. 그림 7은 전체 서비스 관리자의 클래스 다이어그램을 나타낸 것이다.

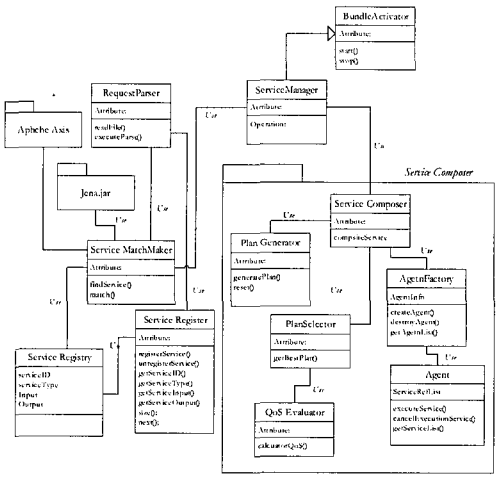


그림 7. 서비스 관리자의 클래스 다이어그램
Fig 7. Class Diagram of Service Manager

작업을 수행할 수 있도록 "Move" 요청을 전달한다. 이를 위해서 미들웨어간 이동 가능한 서비스는 "Mobility" 인터페이스를 상속받아서 구현하도록 한다.

"Mobility" 인터페이스는 미들웨어 이동 전에 수행해야 할 작업들과 다른 미들웨어로 이동 후에 수행해야 할 작업들을 작성하기 위해서 beforeMoving(), afterMoving()이라는 2개의 메소드로 구성된다.

이동 번들 관리기는 서비스 Serializer를 통해서 이동 서비스는 서비스 직렬화 XML을 생성하고, 생성된 XML, 서비스 관리자로부터 얻은 해당 서비스의 등록 정보 내용와 서비스 클래스가 위치한 URL 정보를 이용해서 SOAP메시지로 생성된 후 네트워크를 통해서 전송된다.

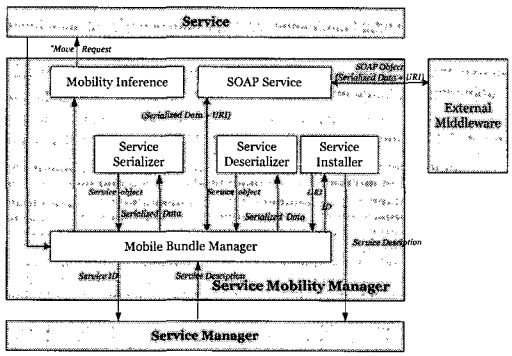


그림 8. 서비스 이동 관리자 구성도
Fig 8. Architecture of Service Mobility Manager

4.3 서비스 이동 관리자

서비스 이동 관리자(service mobility manager)는 상황 인식 서비스가 사용자 등의 이동에 따라서 미들웨어간 이동을 지원한다. 이때, 미들웨어 사이를 이동하는 개체는 미들웨어상의 서비스에 필요한 여러 리소스등을 포함하는 번들 단위로 처리하도록 한다.

서비스 이동 관리자는 서비스와 부가 파일이나 정보로 구성된 번들의 이동을 책임지는 이동 번들 관리자와 서비스 직렬자/역직렬자, SOAP(Simple Object Access Protocol) 서비스, 서비스 설치기로 구성된다. SOAP 은 HTTP 프로토콜 기반해서 XML 문서의 송수신을 가능하게 하며, 호환성 및 확장성이 우수하다. 이동 번들 관리기는 서비스의 "Move" 요청에 따라서 필요한 작업들을 순차적으로 수행하게 된다. 먼저, 서비스에게 현재 수행되고 있는 작업들을 중지를 포함해서 다른 미들웨어상으로 이동하기 전에 필요한

SOAP 메시지를 전송받은 미들웨어는 해당 SOAP 메시지내에 포함된 URL에서 해당 서비스 클래스를 다운로드한 후, 이를 미들웨어의 서비스 관리자를 이용해서 설치하고 실행한다. 이때, 서비스 등록 정보는 서비스 관리자에게 전달해서 서비스 레지스트리에 등록과정에 이용되며, 서비스 직렬화 XML은 역직렬화된 후 서비스의 상태 정보 변경에 이용되어서 설치된 서비스가 이동전의 상태로 복원되도록 한다. 그림 9는 서비스 이동 관리자를 위한 클래스 다이어그램이다.

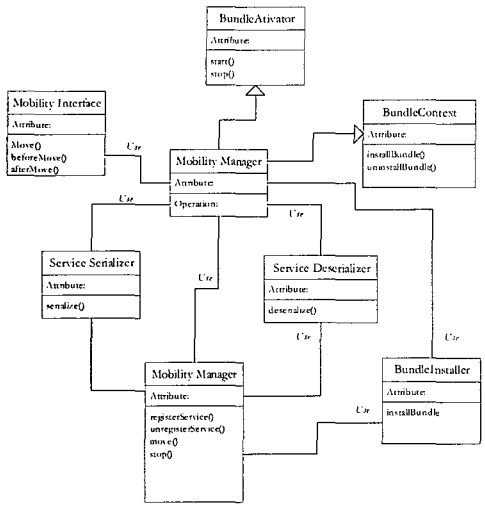


그림 9. 서비스 이동 관리자의 클래스 다이어그램
Fig 9. Class Diagram of Service Mobility Manager

V. 실험 및 평가

이 장에서는 이 논문에서 설계한 동적 서비스 관리 모델에 기반한 상황 인식 미들웨어의 기능과 성능을 실험 및 평가하기 위해서 스마트 홈 네트워크 서비스를 구현하고 구현된 모델의 유효성을 확인하기 위해 서비스의 검색 및 조합, 서비스의 이동성 지원에 대한 실험을 진행하였다. 홈 네트워크 서비스 환경에 사용된 게이드웨이 서버는 Intel P4 2.4GHz, 메모리 1GB, 윈도우즈 서버 2003 환경에서 OSGi 프레임워크인 Knopflerfish 1.3.5를 사용하고, HP의 시멘틱 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.

첫 번째 실험 시나리오는 서비스의 검색 및 조합이 제대로 동작하는지 확인하기 위해 RFID 리더로 확인 된 사용자에 대한 선호도와 상황정보를 기반으로 자동적인 서비스 요청이 이루어지고 알맞은 서비스를 발견하고 서비스 하는지를 평가한다. 실제 가정환경에서 제공될 수 있는 서비스는 그 종류가 많기 때문에 본 논문에서는 상황 인식 제어 서비스를 아래와 같이 커튼, 거실 전등, 침실 전등, 선풍기 및 환풍기 제어로 한정하여 가정환경을 구성하였다.

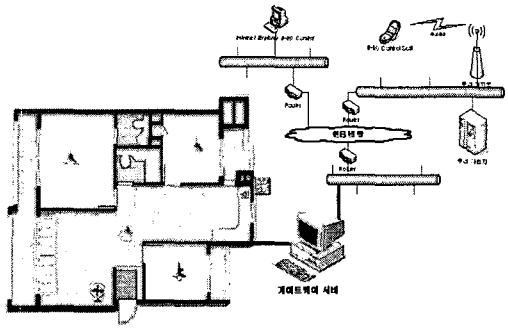
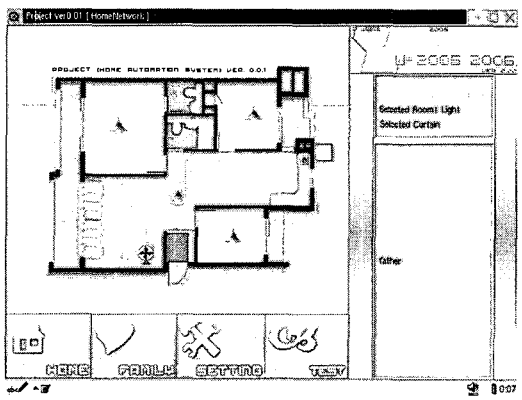
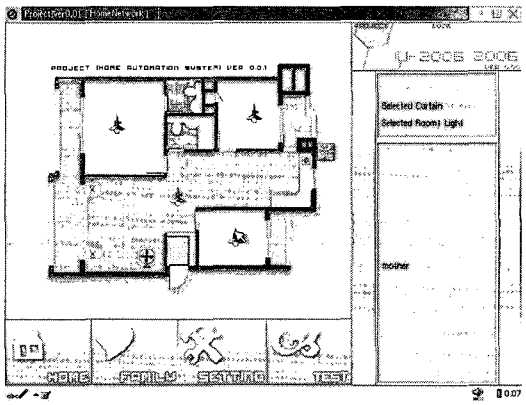


그림 10. 상황 인식 서비스 실험 환경
Fig 10. Experimental Environment

그림 11(a)는 아버지가 귀가 했을 때 아버지의 상황 정보 및 선호도에 따라 커튼을 닫고, 거실 및 방의 전등을 모두 On하는 서비스가 실행되는 것을 보여주고 있다. 그림 11(b)는 어머니의 상황 정보 및 선호도에 따라 환기서비스(커튼 Open,, 환풍기 On)를 실행하고 방의 전등을 모두 Off하는 서비스가 실행되는 것을 보여주고 있다.



(a) 아버지의 상황 정보 및 선호도에 따른 실행



(b) 어머니의 상황 정보 및 선호도에 따른 실행

그림 11. 사용자에 따른 서비스 실행 화면
Fig 11. Service execution screenshot of each person

두 번째로 서비스의 이동성을 실험하기 위해서 MP3 파일 목록과 파일을 포함하고 사용자에게 미들웨어 상의 MP3 파일의 Play를 제공하는 오디오 서비스를 구현하였다. 이를 위해서, PDA 단말기에 미들웨어의 서비스 이동 관리자와 OSGi 프레임워크를 설치하였다. 이 실험에서 사용한 PDA는 HP iPAQ Pocket PC h2210 모델(400MHz Intel XScale 프로세서, 64MB Ram)이며 무선 랜카드를 장착하였다. 이 논문에서 설계한 미들웨어와 Knopflerfish는 IBM J9 컴파일러, Knopflerfish와 미들웨어를 PDA에 포팅하였다. 사용자는 PDA를 통해서 MP3를 듣고 있다가 집안으로 들어오면, PDA 상의 미들웨어가 집안의 미들웨어로 이동하고, 집안에서 MP3 실행이 가능한 PC상의 서비스를 찾아서 플레이할 수 있다.

이 응용 구현을 통해서 PDA상의 미들웨어에서 MP3 파일을 플레이하던 서비스가 서비스 상태를 가지고 PC상의 미들웨어로 이동해서 음악 서비스를 연속되게 서비스 해주었다. 이를 통해서 이 논문에서 제안한 서비스 이동 관리자가 서비스의 현재 상태 정보를 함께 이동시킴으로써 서비스의 이동성을 이상 없이 지원함을 알 수 있었다.

VI. 결론

이 논문에서는 유비쿼터스 컴퓨팅 환경에서 상황 인식 응용을 위한 동적 서비스 관리 모델을 제시하였고, 제시한 모델을 이용해서 상황 인식 서비스 미들웨어를 설계 및 구현하였다. 설계한 미들웨어는 상황 인식 서비스가 동적으로 필요로 하는 서비스를 조합 및 이동할 수 있도록 함으로써 상황 인식 서비스별로 개별화된 상황 정의를 지원하고 기존 연구에서 미흡했던 동적 서비스 지원을 보완했다. 또한, 설계된 미들웨어는 센서와 가전 등의 물리적 기기와 연결하기 위해서 OSGi 프레임워크 환경에서 구현하였다.

이 논문에서 설계한 모델의 타당성을 입증하기 위해서 스마트 홈의 상황을 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 리소스와 대역폭의 제한으로 정확한 상황 정보를 획득하기 어려운 모바일 환경에서의 상황 인식 서비스를 지원하기 위한 프레임워크에 대한 연구이다.

참고문헌

[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] S. A. N. Shafer, B. Brumitt, and J.J. Cadiz, "Interaction Issues in Context-Aware Intelligent Environments," Human-Computer Interaction, Vol. 16, pp. 368-378, 2001.

[8] L. Yan and K. Sere, "A Formalism for Context-Aware Mobile Computing," Proceedings of the Third International Workshop on Parallel and Distributed Computing, Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, pp. 14-21, 2004.

[9] L. Szumel, J. LeBrun, and J. D. Ownes, "Towards a MobileAgent Framework for Sensor Networks," Proceedings of the Second IEEE Workshop on Embedded Networked Sensors, pp. 79-88, 2005.

[10] T. Iwao, S. Amamiya, G. Zhong, and M. Amamiya, "Ubiquitous Computing with Service Adaptation using Peer-to-peer Communication Framework,"Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems, pp. 240-248, 2003.

[11] 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.

[12] 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.

[13] 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.

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

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

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

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

[18] 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.

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

[20] 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

[21] W3C, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl , 2001

저 자 소 개

정 헌 만

1996년 2월 서울산업대학교 전자계산공학과

2001년 2월 인하대학교 전자계산공학과 (공학석사)

2004년 2월 인하대학교 컴퓨터정보공학과 (박사수료)

2001년~ 경인여대,인하전문대 현재 강사

관심분야 상황인식, 시맨틱웹, 웹서비스, 유비쿼터스 센서네트워크

이 정 현

1977년 2월 인하대학교 전자공학과

1980년 2월 인하대학교 전자공학과 (공학석사)

1988년 2월 인하대학교 전자공학과 (공학박사)

1979년~ 한국전자기술연구소 1981년 시스템 연구원

1984년~ 경기대학교 1989년 전자계산학과 교수

1989년~ 인하대학교 현재 컴퓨터공학부 교수

관심분야 HCI, 유비쿼터스 컴퓨팅