퍼베이시브 홈 환경을 위한 온톨로지 기반의 시멘틱 서비스 탐색 기법

(An Ontology-based Semantic Service Discovery Scheme for Pervasive Home Network Environments)

조 미 영 † 강 세 훈 †† 이 영 희 †††
(Miyoung Cho) (Seahoon Kang) (Younghee Lee)

요 약 서비스 탐색기법은 퍼베이시브 환경을 지향하는 홈 네트웍 환경에서 중요한 기술 중 하나로 연구되고 있다. 그러나 기존의 연구들은 디바이스나 서비스가 갖는 시멘틱을 이해하지 못하고 단순히 구문 검색에 의한 매칭기법만 제공하고 있다. 본 연구에서는 이와 같은 문제점을 해결하기 위하여 시멘틱 서비스 탐색을 위한 온톨로지를 개발하였다. 개발된 온톨로지는 시멘틱을 포함하여 퍼베이시브 홈 네트워크 환경 내의 디바이스나 서비스를 기술하고, 이들의 관계를 프리미티브 서비스 개념을 통해 효과적으로 기술하였다. 또한, 디바이스에 대한 프로퍼티를 표현하여, 퍼베이시브 환경에서 디바이스가 갖는 여러 가지 위치 정보나 디바이스 상태 등의 정보가 기술되도록 하였다. 본 논문에서는 이와 같이 정의된 온톨로지를 사용하여 서비스를 검색하고, 그 유용성을 평가하기 위해 기존의 Jini 룩업서비스를 확장하여 시멘틱 서비스 탐색 시스템을 개발하였다. 그리고 개발된 시스템에서 다양한 시나리오상의 서비스 탐색 실험을 통해, 온톨로지에 의한 시멘틱 탐색기법의 유용성을 입증하였다.

키워드 : 시멘틱 서비스 탐색기법, 온톨로지

Abstract In recent years, service discovery is one of the major technologies of home networks which head for a pervasive computing environment. However, existing service discovery techniques are difficult to understand semantics, and they only provide syntactic level service matching. To solve these problems, we have designed and developed ontology for semantic service discovery. Our ontology could enrich the amount of devices and services representations with semantics, and the relation of devices and service could be efficiently described through primitive service. For representing context information of devices, we describe attributes of device including location information, device status and etc. To determine whether the developed ontology can be applied to service discovery systems, we have implemented a semantic service discovery system by extension of the existing Jini lookup service. Also, we have evaluated our ontology with associated software environment according to some experiment scenarios, and have proved the usefulness of our ontology-based semantic service discovery system.

Key words : semantic service discovery, ontology

1. 서 론

컴퓨터의 보급과 인터넷의 확산에 의하여 정보의 전달과 획득이 가정에서도 일상화 되었고, 디지털 가전 및 각종 정보 기기의 개발에 따라 이를 네트워크로 연결하여 이를 가정 안팎에서 제어하고자 하는 홈 네트워킹에 대한 연구가 활발하게 진행 중이다. 이러한 홈 네트워킹 연구는 언제 어디서나 이용할 수 있고, 사람과 컴퓨팅 디바이스 및 환경이 서로 상호 작용할 수 있는 퍼베이시브 환경을 추구하고 있다. 퍼베이비스 환경을 지향하는 홈 네트워킹에 대한 연구가 활발히 진행되면서 서비스 탐색 기법에 관한 중요성이 더더욱 증대되고 있다. 서비스 탐색 기법은 컴퓨팅 디바이스들이 관리자 없이도 자동으로 네트워크 설정이 이루어지고, 사용자가 원

† 비 회 원 : 삼성전자 정보통신총괄 무선사업부 연구원
miyoung79.cho@samsung.com
†† 비 회 원 : 한국정보통신대학교 공학부
kang@icu.ac.kr
††† 종신회원 : 한국정보통신대학교 공학부 교수
yhlee@icu.ac.kr
논문접수 : 2004년 1월 30일
심사완료 : 2005년 1월 19일

하는 서비스에 대한 요청 메시지를 보냈을 때, 서비스 요청에 부합하는 서비스를 자동으로 검색하여 찾아주는 기법이다.

이와 같은 현재의 서비스 탐색기법은 크게 두 가지로 분류된다. 먼저 UPnP[12]나 SLP[13], Jini[11]과 같이 속성 이름 기반(attribute-based)의 검색하는 기법과 UDDI[14], CORBA Naming Service[15], RMI Registry [16]와 같이 인터페이스 이름(interface-name) 기반의 검색기법이 있다. 이와 같은 기존의 서비스 탐색 기법은 다음과 같은 몇 가지 제한점을 갖는다. 먼저 서비스의 기본적인 정보만 기술되기 때문에 서비스의 기능 및 특징에 대해 파악하기가 힘들고, 서비스를 이용하기 위해 인터페이스에 관한 자세한 정보까지 알아야 한다. 또한 현재의 방법으로는 질의에 정확히 매칭되는 서비스만을 검색할 수 있고, 부정확한 매칭(inexact matching)에 대한 고려가 없다. 본 논문에서는 단순한 구문 매칭이 아닌 의미를 포함한 부정확한 매칭을 지원하기 위하여 온톨로지를 이용하여 서비스를 기술하고, 서비스를 탐색하는 과정에서 온톨로지에 기술된 내용과 추론엔진을 사용하여 사용자가 원하는 서비스가 검색되어 지도록 한다.

온톨로지란 공유된(shared) 개념들의 개념화(conceptualization)를 정형적(formal)이고 명시적으로(explicit) 설명한 명세서(specification)를 의미한다.[2,3] 다시 말해 온톨로지는 특정 도메인에서, 정보를 공유하기 위해 관련된 단어와 이들의 관계를 규정하고, 이를 프로그램이 이해할 수 있도록 한 것이다. 따라서 이러한 온톨로지를 이용하여 가정환경 내의 디바이스/서비스를 기술하면, 가정 내의 각 디바이스와 서비스들의 자세한 정의가 가능하고, 이들 사이의 계층분류와 관계정의를 하는데 용이하다. 또한, 온톨로지를 이용한 추론이 가능하기 때문에 직접적으로 표현되지 않은 개념에 대한 사용자의 요청 메시지가 들어왔을 때, 개념간의 관계 추론을 통하여 요청 메시지에 응답할 수 있다. 뿐만 아니라, 같은 개념에 대해 서로 다른 식별자를 사용할 경우에 온톨로지를 이용하여 이들의 의미를 맞춰줄 수 있다. 예를 들어, 제조업체에 대한 식별자로 MANUFACTURER와 MAKER 두 가지를 사용할 경우, 이들이 의미하는 대상이 같다는 것에 대한 정의가 온톨로지에 정의 되어 있기 때문에, 의미를 혼동하지 않고 사용할 수 있다. 본 논문에서는 이와 같은 장점을 갖는 온톨로지 기반의 서비스 탐색기법을 제안하고, 이를 위해 가정환경에 특화된 온톨로지를 개발한다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서 기존의 서비스 탐색 기법에 대해 분석하고, 3장에서는 가정환경에 특화된 온톨로지를 개발하고, 개발한다. 4장에서는 온톨로지를 이용한 서비스 탐색을 위한 시스템 모델을 설명하고, 시스템 아키텍처를 그린 후 구현을 위한 세부사항에 대한 설명을 한다. 5장에서는 구현된 온톨로지와 전체 시스템에 대한 평가를 하고, 마지막으로 6장에서 결론을 내린다.

2. 관련연구

2.1 시멘틱 서비스 탐색연구

시멘틱 서비스 탐색 기법은 출발점을 시멘틱 웹[4]에서 시작한다. 시멘틱 웹은 기존의 웹상에 있는 정보에 의미를 부여하여 사람 뿐 아니라 컴퓨터도 정보에 대한 의미를 해석할 수 있도록 한 기법이다. 따라서 기존의 웹 문서와 달리, 자동화된 에이전트나 검색엔진들이 웹 문서에 부여된 의미를 해석할 수 있고, 이로 인해 자동화되고 지능화된 웹 서비스의 검색이 가능하게 되었다. 시멘틱 웹에서는 의미를 부여하여 웹 문서를 기술하기 위하여 온톨로지를 이용하고, 사용자가 원하는 웹 문서를 검색할 때 추론엔진을 이용하여 원하는 결과를 얻을 수 있게 하였다. 대부분의 시멘틱 서비스 탐색기법도 시멘틱 웹과 비슷하게 서비스를 기술하기 위하여 온톨로지를 이용하고, 원하는 쿼리에 가장 근접한 결과를 만들기 위하여 추론엔진을 이용한다.

Dreggie[5]는 M-commerce 어플리케이션을 위한 시멘틱 서비스 탐색기법을 제안한 시스템이다. 여기에서는 모바일 서비스를 기술하기 위하여, 서비스들의 기능(functionality)이나 성능(capability), 플랫폼 요구사항(platform requirements) 등을 온톨로지로 나타내었다. 또한 서비스 매칭을 위하여, 기존의 SUN사에서 개발한 Jini lookup/ discovery 메커니즘에 프롤로그(prolog)엔진을 추가하여 보다 복잡한 매칭이 가능하게 하였다.

Enhanced Bluetooth Service Discovery Protocol[6]는 기존의 블루투스 네트웍에서 Universally Unique Identifiers(UUID)를 기반으로 서비스 탐색을 하는 메커니즘을 개선한 방법이다. 기존의 서비스 탐색기법에서는 매우 제한된 서비스 기술만 가능하고, 서비스 요청에 대한 응답으로 yes/no형태만 가능하였다. 이를 개선하기 위하여, 이 시스템에서도 마찬가지로 서비스를 표현하기 위해서 DAML(DARPA Agent Markup Language)로 서비스 온톨로지를 기술하고, 프롤로그 엔진을 이용해 서비스 요청에 대한 결과 값을 얻어내었다.

위의 시스템들은 퍼베이시브 홈 네트웍 환경이 아닌 한정된 도메인에서 서비스들을 기술하여, 디바이스나 서비스의 다양한 컨텍스트를 기술하지 못한다. 또한, 가변적인 변수들을 온톨로지에 나타내었을 경우, 이 값들을 계산하여 적절한 서비스를 찾아내는데 제한점을 갖는다.

2.2 Pervasive computing 환경을 위한 온톨로지

온톨로지는 퍼베이시브 환경에서 지식공유(knowledge sharing)와 표현을 위한 주요 기술 중 하나이다. 따라서 퍼베이시브 환경을 지향하는 여러 시스템들은 온톨로지를 개발하여 사용하고 있다. 본 논문에서는 CoBrA시스템과 GAIA시스템에 대해 분석한다.

CoBrA시스템은 인텔리전트 미팅룸을 도메인으로 하여 COBRA-ONT라는 온톨로지를 정의하였다[8]. COBRA-ONT는 owl로 개발되었으며, 장소(place), 에이전트(agent), 이벤트(event)등에 연관된 개념을 포함하고 있다. 예를 들어, 장소에 연관된 클래스들은 Campus, Building, Room등과 같이 물리적인 장소를 의미하는 클래스를 포함하고, 에이전트에 연관된 클래스들은 휴먼 에이전트나 소프트웨어 에이전트를 포함한다. 추가적으로, COBRA-ONT에서는 PersonInMeetingRoom, PresentationHappeningNow 등과 같이 에이전트가 있는 장소나 에이전트가 수행중인 활동을 표현하기 위한 클래스들을 정의하였다. 이와 같이, COBRA-ONT는 에이전트의 컨텍스트를 표현하는데 초점이 맞춰져있고, 장소나 활동 등에 대한 관계를 나타내기 위해 추가적인 클래스들을 정의함으로써 이를 표현하였다. 그러나 관계를 나타내기 위해 추가적인 클래스를 정의하게 되면, 도메인의 범위가 넓어지고 에이전트의 종류가 많아질수록 온톨로지의 복잡성(complexity)이 증가하게 된다.

GAIA시스템에서도 퍼베이시브 환경의 다양한 객체를 표현하고, 이들의 컨텍스트 정보를 표현하기 위해 온톨로지를 개발하였다[7]. 정의한 객체는 서비스(Service)나 디바이스(Device), 어플리케이션(Application), 사용자(User) 등을 포함하고, 컨텍스트 정보를 나타내기 위해 위치(Location), 활동(Activity), 날씨정보(Weather Information) 등을 포함한다. 이와 같이 GAIA에서 정의한 온톨로지는 퍼베이시브 환경의 여러 객체를 표현하고 있다. 하지만, 각 개체의 의미를 효과적으로 표현하지 못한다. 예를 들면, 모바일 폰은 전화 서비스 뿐만 아니라 디스플레이 서비스, 사운드 서비스, 스케줄러 서비스, 카메라 서비스 등을 제공할 수 있다. 이와 같이 제공 가능한 서비스를 나타내기 위해서 디바이스 온톨로지는 각 서비스와의 관계를 일일이 기술하여야 한다. 이는 해당 도메인의 범위가 넓어져 서비스와 디바이스의 종류가 더욱 많아지고, 이들의 관계가 늘어날수록 온톨로지의 복잡성가 증가한다. 또한, 새로운 서비스가 정의될 때에는 기존 디바이스 온톨로지를 수정하여야 하기 때문에 비효율적이다.

3. 시멘틱 서비스 탐색을 위한 온톨로지

3.1 디자인 고려 사항

본 논문에서 온톨로지 개발 시 고려해야할 사항은 다음과 같다.

- 온톨로지는 집안환경에 있는 각종 디바이스와 서비스를 모두 표현할 수 있어야 한다. 이때, 각 디바이스 및 서비스의 종류 뿐 아니라 이들의 관계, 가지고 있는 특성이나 기능들에 대해서도 모두 표현이 가능하여야 한다.
- 원하는 디바이스를 검색하고 난 후, 해당 디바이스를 제어하기 위하여 디바이스의 상태변수와 컨트롤 인터페이스에 대한 정보도 온톨로지에 함께 기술되어야 한다.
- 같은 종류, 같은 기종의 디바이스가 집안환경에 추가될 경우에, 이들이 서로 구분될 수 있어야 한다.
- 이렇게 제안된 온톨로지는 새로운 가전 디바이스나 서비스가 더해지거나, 혹은 새로운 용어가 정의되더라도 기존의 온톨로지를 수정하지 않고 수용할 수 있는 확장성을 가져야 한다.

3.2 온톨로지 개발 방법

온톨로지를 개발하는데 있어서 약속된 절차나 개발 방법이 있는 것은 아니다. 적절한 온톨로지의 개발은 타겟이 되는 어플리케이션이나 해결하고자 하는 문제를 고려하고, 계속적인 수정과 세부항목을 채워나가는 보완 작업을 통해 이루어진다. 본 논문에서는 'Ontology Development 101'[9]에서 소개한 개발절차에 따라 온톨로지를 개발한다.

온톨로지를 개발하는데 있어 또 하나 중요한 것은 어떤 언어를 사용하여 온톨로지를 기술할지를 정하는 것이다. 온톨로지를 기술하는 언어로는 XML, RDF, DAML+OIL, OWL 등이 있다. 이중 본 논문에서는 DAML+OIL을 이용하여 온톨로지를 기술한다. DAML+OIL은 온톨로지를 클래스와 프로퍼티로 표현을 하고, 이들의 명확한 정의, 클래스와 클래스의 관계, 프로퍼티와 프로퍼티 간의 관계를 규정하는 방법을 제공한다. 즉, 기존의 XML과 XML 스키마, RDF와 RDF 스키마를 확장, 강화하면서, 동시에 디스크립션 로직에서의 표현들이 모두 가능하다. 때문에 온톨로지에 직접적으로 기술되지 않은 개념들에 대해서도 추론을 통한 응답이 가능하다.

3.3 목적 및 도메인 분석

본 논문에서 제안하는 온톨로지는 퍼베이시브 홈 네트워크 환경 내에 있는 모든 디바이스에 의미를 부여하여 기술하는 것을 목적으로 한다. 이때 디바이스는 일반 중산층 가정에서 현재 사용되는 것들로 제한되며, 이외의 것들에 대해서는 고려되지 않는다. 정의된 온톨로지는 각 디바이스들이 가지고 있는 모델이름, 제조업체 등과 같은 기본적인 애트리뷰트를 포함한다. 또한, 각 디바이스들이 제공하는 프리미티브 서비스들에 대한 정보

를 독립적으로 유지하면서, 이들이 가지고 있는 상태변수나 컨트롤 인터페이스에 대한 정보를 포함한다. 다시 말해, 서비스를 제공하는 물리적인 개체인 각 디바이스에 대한 정보와 실제 서비스를 제공하는 논리적 단위인 프리미티브 서비스에 대한 정보를 포함한다. 또한, 프리미티브 서비스의 조합이나 프리미티브 서비스와 디바이스 상태의 조합으로 이루어진 서비스에 대한 정보도 포함한다.

3.4 온톨로지 구조

본 장에서는 제안하는 온톨로지의 구조에 대해 설명하고, 각 온톨로지의 관계 및 세부사항에 대해 설명한다.

그림 1은 본 논문에서 제안하는 온톨로지의 계층구조를 나타낸 것이다. 엔터티 클래스는 최상위 클래스로써, 디바이스/서비스 등을 포함한 모든 객체는 엔터티 클래스의 하위 클래스가 된다. 점선에 해당하는 것은 프로퍼티를 나타낸다.

먼저 서비스 탐색을 위한 주요 요소인 디바이스 클래스와 서비스 클래스를 정의하고, 디바이스를 표현하기 위하여 디바이스의 프로퍼티로 애트리뷰트 클래스와 프리미티브 서비스 클래스를 레인지(range)로 갖는 프로퍼티를 정의하였다. 애트리뷰트 클래스는 퍼베이시브 환경에서 디바이스가 갖는 여러 가지 프로퍼티를 표현하기 위한 것으로, 위치 정보와 같은 가변적인 속성도 여기에 포함된다. 프리미티브 서비스 클래스는 디바이스가 가지고 있는 최소한의 서비스들을 표현한 것으로, 이들의 조합으로 서비스 클래스가 구성된다. 만약, 디바이스 클래스와 서비스 클래스의 매개체 역할을 하는 프리미티스 서비스 클래스를 사용하지 않는다면, 각 디바이스가 제공할 수 있는 서비스를 각각 기술하여야 하기 때문에 매우 비효율적이고 온톨로지에 복잡성이 증가한다. 따라서 우리는 프리미티브 서비스 클래스를 통해 디바이스 클래스와 서비스 클래스간의 관계를 효과적으로 나타내고, 디바이스가 제공할 수 있는 서비스들을 시멘틱하게 모두 표현하였다. 디바이스가 검색되고 난 후에 실제로 디바이스를 이용하기 위해서는 컨트롤을 하기 위한 인터페이스 이름이나 상태변수들을 알아야 한다. 이를 위해 프리미티브 서비스 클래스는 상태변수 클래스와 컨트롤 인터페이스 클래스를 레인지로 갖는 프로퍼티를 갖는다. 각 클래스에 대한 세부 구조 및 설명은 다음과 같다.

3.4.1 디바이스 클래스(Device class)

디바이스는 가정환경에서 서비스를 제공하는 최소의 물리적인 단위이다. 본 논문에서는 가정 내의 디바이스들을 디바이스 클래스로 표현하기 위하여, 먼저 디바이스를 사용용도에 따라 크게 9가지로 분류한다. 그리고 다시 각 분류마다 세부 분류를 하여 디바이스를 계층적으로 분류한다.

각 디바이스는 독립적으로 자신의 모델명이나 제조일자 등과 같은 스테틱 애트리뷰트(static attribute)와 위치정보나 디바이스 상태 등과 같은 다이나믹 애트리뷰트(dynamic attribute)를 갖는다. 또한 각 디바이스에 속해있는 실제 서비스를 제공하는 프리미티브 서비스와 하위 디바이스로 포함하는 임베디드 디바이스들에 대한 정보가 기술된다.

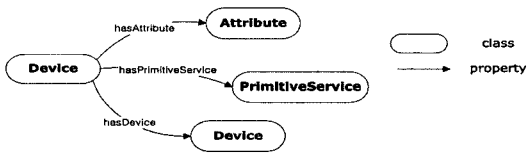

그림 2 디바이스 클래스의 구조

그림 2와 같이 디바이스 클래스는 hasAttribute, hasPrimitiveService, hasEmbeddedDevice의 프로퍼티를 갖는다. hasAttribute는 디바이스가 갖는 애트리뷰트를 나타내기 위한 것으로, 레인지는 애트리뷰트 클래스를 갖는다. hasPrimitiveService는 디바이스가 갖는 프리미티브 서비스를 기술하기 위한 프로퍼티로, 레인지 값은 프리미티브 서비스클래스가 된다. 마지막으로

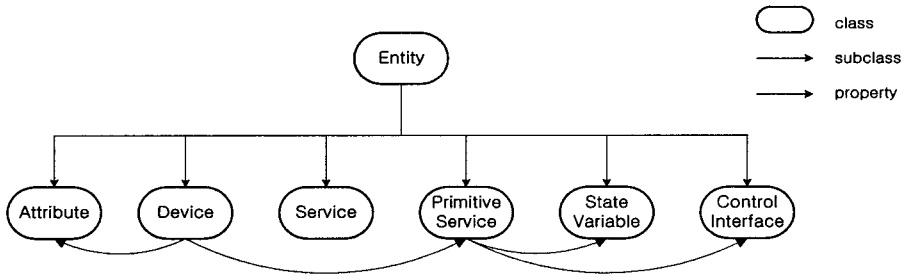

그림 1 상위레벨의 클래스

hasEmbeddedDevice는 디바이스가 갖는 임베디드 디바이스를 기술하기 위한 것으로, 레인지 값은 디바이스 클래스가 된다.

3.4.2 프리미티브 서비스 클래스(Primitive service class)

프리미티브 서비스는 서비스를 제공하는 최소의 논리적 단위로써, 디바이스가 갖는 서비스를 표현하면서 동시에 서비스를 구성하는 요소가 된다. 즉, 디바이스와 서비스를 연결하는 매개체 역할을 한다. 프리미티브 서비스 클래스에는 각 프리미티브 서비스가 갖는 상태변수와 컨트롤 인터페이스에 대한 정보가 기술된다.

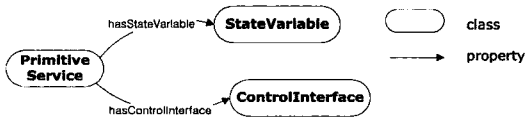

그림 3 프리미티브 서비스 클래스의 구조

그림 3과 같이 프리미티브 서비스 클래스는 has StateVariable과 hasControlInterface의 프로퍼티를 갖는다. hasStateVariable은 프리미티브 서비스가 갖는 상태변수들을 나타내기 위한 것으로, 레인지로는 상태변수 클래스를 갖는다. hasControlInterface는 프리미티브 서비스가 갖는 인터페이스들을 기술하기 위한 것으로, 레인지 값은 컨트롤 인터페이스 클래스가 된다.

3.4.3 서비스 클래스(Service class)

서비스는 앞에서 언급한 프리미티브 서비스와 의미가 구분된다. 본 논문에서 제시하는 서비스는 보통 사용자들이 인식하고 있는 디스플레이 서비스나 알람 서비스와 같은 서비스를 말하는 것으로, 이것은 각 프리미티브 서비스들의 조합이나 프리미티브 서비스와 디바이스 애트리뷰트의 조합으로 이루어질 수 있다. 서비스 클래스에는 이와 같은 내용이 기술된다.

그림 4는 가능한 서비스의 조합을 예를 들어 나타낸 것이다. 먼저, 프리미티브 서비스의 조합은 하나의 서비스를 구성하게 된다. 예를 들어, 알람 서비스는 알람 프리미티브 서비스 하나만으로 알람 서비스의 기능을 할 수 있고(a), 클락 프리미티브 서비스와 사운드 프리미티브 서비스가 합쳐져서 알람 서비스의 기능을 할 수 있다(b). 다음으로, 프리미티브 서비스와 디바이스의 애트리뷰트의 조합으로 또 하나의 서비스를 표현할 수 있다. 예를 들어, 디스플레이 프리미티브 서비스와 라지 스크린이라는 디바이스의 애트리뷰트가 합쳐지면 라지 스크린 서비스를 나타낼 수 있다.

3.4.4 애트리뷰트 클래스(Attribute class)

애트리뷰트 클래스는 디바이스의 특성을 나타내는 클래스로 디바이스 클래스의 프로퍼티인 hasAttribute의 레인지 값이 된다. 하위 클래스로는 값이 변하지 않고 일정한 값을 갖는 스테틱 애트리뷰트 클래스와 위치 정보와 같이 값이 계속해서 변하는 다이나믹 애트리뷰트 클래스가 있다. 다시 하위 클래스는 ModelName, ModelNumber, SerrialNumber, Manufacturer, Making-Date 등과 같이 대부분의 디바이스들이 모두 지니는 커먼 애트리뷰트(Common Attribute) 클래스와 Screen-Size, ColorSupport와 같이 디바이스 스페시픽(Device-specific Attribute) 애트리뷰트 클래스로 나뉜다.

3.4.5 상태변수 클래스(State Variable class)

상태변수란 프리미티브 서비스가 가지고 있는 각각의 변수를 나타내는 것으로 프리미티브 서비스 클래스가

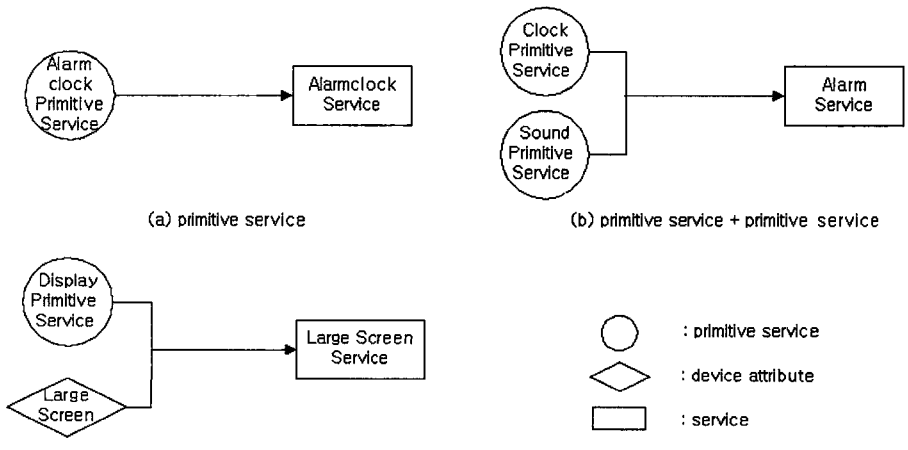

그림 4 서비스 클래스의 구조

갖는 hasPrimitiveService의 레인지가 된다. 예를 들어, 사운드 프리미비트 서비스의 경우에는 볼륨이나 밸런스 등을 상태변수로 가지게 된다. 이와 같은 상태변수는 서비스를 제어할 때 컨트롤 인터페이스에 들어가는 파라미터 값이 된다. 상태변수 정보를 온톨로지에 포함시킴으로써, 원하는 서비스를 검색한 후 서비스를 이용할 때, 변수의 이름이 각 서비스마다 다르게 정의되어 있을 때 의미를 맞추어 사용할 수 있다.

3.4.6 컨트롤 인터페이스 클래스(Control Interface class)

원하는 서비스를 검색하고 난 후에 디바이스를 제어하기 위해서는 해당 컨트롤 인터페이스를 알아야 한다. 이를 위해 컨트롤 인터페이스 클래스를 정의한다. 상태변수와 마찬가지로 컨트롤 인터페이스는 프리미티브 서비스 클래스가 갖는 hasPrimitiveService의 레인지가 되고, 이러한 정보를 온톨로지에 포함시킴으로써, 인터페이스 이름이 서비스마다 각기 다르게 정의하였더라도 의미를 맞추어 사용할 수 있다.

3.5 온톨로지 구현

위에서 디자인한 온톨로지는 온톨로지 개발도구를 이용하여 작성된다. 일반적으로 개발도구들은 GUI환경에서 개발하는 도구이기 때문에, 시간을 단축할 수 있고, 결과물을 시각적으로 보고 검증할 수 있다. 본 논문에서는 여러 가지 개발도구 중에서 맨체스터 대학에서 개발한 OilEd[10]를 사용하여 온톨로지를 작성하였다. OilEd는 DAML+OIL을 사용하여 온톨로지를 개발할 수 있는 에디터로, 이 개발도구에 추론엔진을 연결하여 일관성 검사를 할 수 있는 기능을 제공한다. 또한, 여러 가지 인터페이스를 제공하기 때문에, 클래스나 프로퍼티 정의와 관계기술이 용이하다.

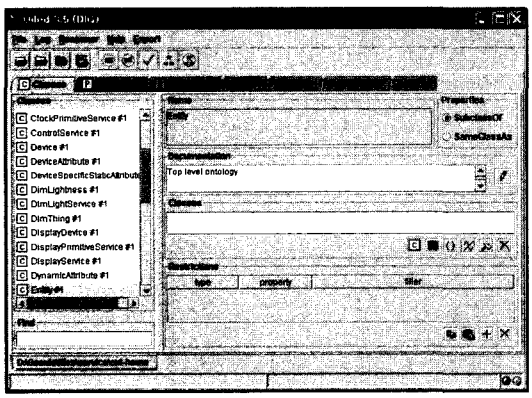

그림 5 OilEd를 사용하여 classes, slots, facets 정의

4. 온톨로지를 이용한 서비스 탐색 시스템 설계 및 구현

4.1 시스템 모델(System model)

본 논문에서 제안하는 시스템 모델은 Jini기반의 서버/클라이언트 모델이다. Jini[11]의 특징은 네트웍 상에서 플러그 앤 플레이를 제공하고, 서비스의 자동 설정 등으로 인해 관리과정을 줄일 수 있다는 것이다. 또한 등록한 서비스들의 목록을 가지고 있는 룩업서비스를 제공하고, 새로운 서비스가 네트웍에 추가되거나 업데이트 되었을 때, 이를 자동으로 감지하여 업데이트 한다. 이와 같은 이유로 본 논문에서는 기존의 Jini시스템 모델을 활용하고, 시멘틱 서비스 탐색을 위하여 Jini의 룩업서비스에 추가적인 컴포넌트를 정의한다.

본 논문에서는 Jini 시스템 모델과 마찬가지로 네트웍 상에 레지스트리의 역할을 하는 룩업서비스를 두고, 클라이언트, 서비스, 룩업서비스 삼자 간에 상호통신이 이루어지도록 한다. 클라이언트와 서비스는 룩업서비스에 등록하는 과정과 검색하는 과정을 거친 후 룩업서비스를 통하지 않고 직접 통신을 할 수 있다. 이때 클라이언트는 Jini와 마찬가지로 인터페이스 이름을 기반으로 one-to-one matching을 하는 것이 아니라, 추가적으로 정의된 컴포넌트를 이용하여 서비스를 검색한다. 추가적으로 정의된 컴포넌트 중 디바이스 디스크립션 리포지토리(Device description repository)는 정의된 온톨로지를 이용하여 기술된 디바이스 정보(instance)를 포함하고 있으며, 여기에는 디바이스가 제공할 수 있는 모든 서비스가 기술되어지도록 하고, 상태변수나 인터페이스 정보까지 기술하여 디바이스가 검색되어지고 난 후에 이를 컨트롤 할 수 있도록 한다. 룩업서비스가 디바이스 디스크립션 정보를 가지고 추론을 하여 사용자가 요청한 서비스를 찾아낼 때에는 정의된 온톨로지를 통해 추론과정이 이루어지게 된다.

4.2 룩업서비스 구조(architecture)

그림 7에서 보는 바와 같이, 제안하는 룩업서비스의 구조는 크게 세 가지로 나뉜다. 먼저, 서비스의 등록을 담당하는 레지스트리가 있고, 시멘틱을 포함하는 정보에 대해서 추론을 통해 결론을 도출할 추론 엔진, 마지막으로 클라이언트의 쿼리 메시지를 해석하고, 쿼리 메시지에 따라 한단계/두단계 검색인지를 구분하는 매칭관리자가 있다. 또한 추가적으로 다이나믹 애트리뷰트에 대한 평가를 담당하는 평가자(Evaluator)와 동적값 추출자(Dynamic-value Extractor)가 있다.

- **레지스트리(Registry)** : 레지스트리는 서비스의 등록을 담당하는 컴포넌트로 서비스가 네트웍에 추가되면, 서비스는 디스커버리(discovery)와 조인(join) 과정을 통해 레지스트리에 등록한다. 레지스트리에 등록되는 것은 각 디바이스가 본 논문에서 정의된 온톨로지로 기술된 디스크립션 파일이 된다.

- **추론엔진(Reasoning engine)** : 추론엔진은 사용자의 쿼리 메시지를 받고, 추론과정을 통해 사용자가 원하는 서비스를 찾는 역할을 한다. 이때 추론엔진은 레지스트리에 등록되어 있는 디바이스의 인스턴스들 중, 정의된 온톨로지를 기반으로 추론과정을 거쳐 사용자가 원하는 서비스를 검색한다. 본 논문에서는 추론엔진으로 RACER(Renamed ABox and Concept Expression Reasoner)[19]를 사용하였다.
- **매칭관리자(Matching Manager)** : 매칭관리자는 사용자의 쿼리 메시지 받아 이를 해석하고, 스테틱 애트리뷰트에 대한 추론만 필요한지, 혹은 추가적으로 다이나믹 애트리뷰트에 대해 평가가 필요한지를 결정하는 역할을 한다.
- **평가자(Evaluator)** : 다이나믹 애트리뷰트에 대해 평가를 하는 컴포넌트로, 온톨로지에 기술된 내용을 받아와 이에 따라 값을 평가한다. 예를 들어 'SameRoom'에 해당하는 서비스들을 추출해야 할 경우, 'SameRoom'에 대한 의미는 온톨로지에 기술된 내용을 바탕으로 해석하고, 실제 이에 대한 계산은 평가자에서 한다. 이때 서비스나 클라이언트의 다이나믹 애트리뷰트의 정보를 받아오기 위하여 동적값 추출자에게 이를 요청한다.
- **동적값 추출자(Dynamic-value Extractor)** : 평가자가 요청한 값을 위치서버(*Location Server*)나 디바이tm에게서 실제로 받아오는 역할을 한다.

4.3 동작 과정(Operation Scenario)

4.3.1 서비스 등록

서비스를 등록하는 과정은 그림 8과 같다. 먼저 서비스는 레지스트리에게 서비스 등록 요청 메시지를 보낸다. 이때 자신의 디스크립션 파일이 어디에 있는지에 대한 URL주소를 함께 보낸다. 이를 받은 레지스트리는 서비스에게 ID를 부여하고, 디바이스에 대해 기술되어있

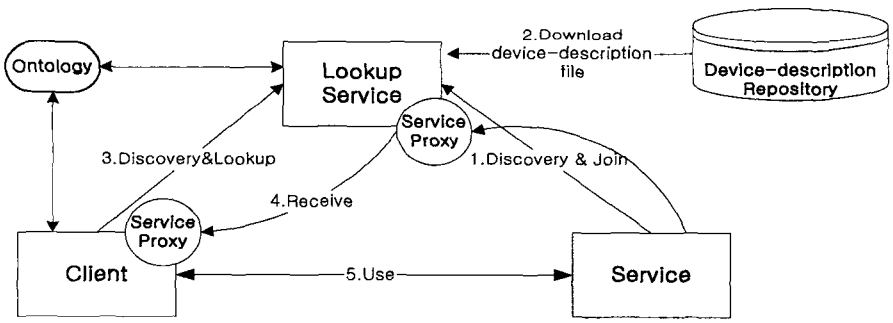

그림 6 시멘틱 서비스 탐색을 위한 확장된 Jini기반 시스템 모델

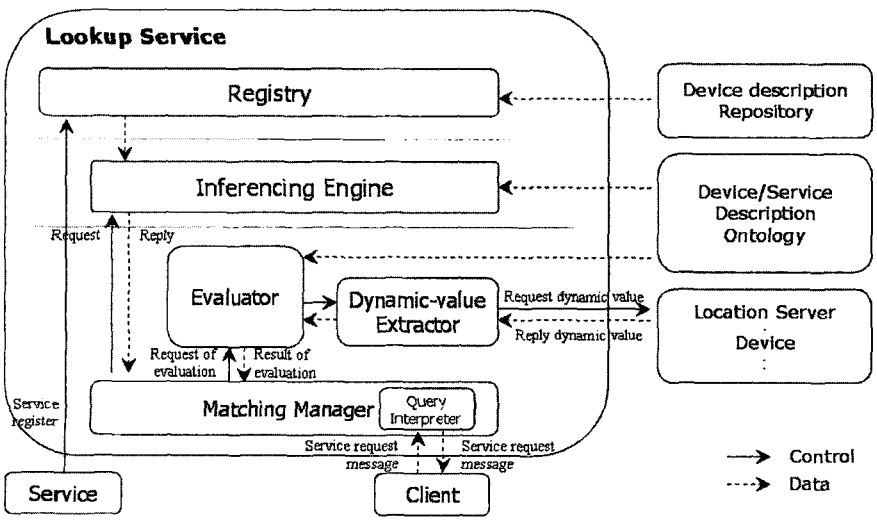

그림 7 룩업서비스 구조

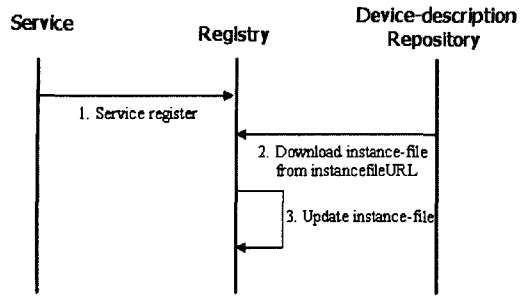

그림 8 서비스 등록과정

는 디바이스 디스크립션 리포지토리에게 디바이스에 대해 기술되어 있는 인스턴스 파일을 요청한다. 해당 파일을 받은 레지스트리는 이를 추가하여 현재의 인스턴스 목록 파일을 업데이트 한다.

4.3.2 서비스 요청 및 응답

클라이언트가 원하는 서비스를 요청하고 이를 응답받는 과정은 그림 9와 같다.

클라이언트가 서비스를 요청하면 쿼리 메시지는 먼저 매칭관리자가 전송된다. 매칭관리자의 쿼리 인터프리터는 쿼리 메시지를 해석하여, real-time바인딩을 통해서 값을 업데이트 시켜야 하는 변수가 있는지를 체크한다. 만약 스테틱 애트리뷰트로만 구성된 쿼리 메시지이면 사용자의 쿼리 메시지를 추론엔진에게 보내 정적인 값을 갖는 부분에 대한 추론결과를 받아오고, 결과 값을 다시 클라이언트에게 전송해준다. 그러나 만약 다이나믹 애트리뷰트가 고려되어야 하는 부분이 쿼리 메시지에 포함되어 있었다면, 추론엔진에서 받은 결과 리스트들에 한해 다이나믹 애트리뷰트에 대한 평가부분이 추가 되어야 한다. 이와 같은 과정이 6~12까지에 해당된다. 온톨로지 내에서는 더하기, 빼기 등의 실제 연산이 불가능하기 때문에, 다이나믹 애트리뷰트의 실제 연산에 관련된 부분은 추론엔진이 아니라 별도의 컴포넌트에 의해 실행된다.

5. 실험 및 평가

5.1 온톨로지 기술적 평가

온톨로지는 온톨로지 개발 툴을 이용하여, 구문(syntax) 검사 및 순환성(circularity), 일관성(consistency)에 대한 검사를 할 수 있다[17]. 본 논문에서는 OilEd에 FaCT[20] 추론엔진을 연결하여 구문에러, 순환성 및 일관성을 검사를 하였다. 다시 말해, 기술된 온톨로지가 문법적으로 잘못된 구조가 없는지, 클래스나 프로퍼티 등을 정의할 때 잘못된 키워드를 사용하지 않았는지에 대한 검사를 하였다. 또한, 개념을 정의할 때 이들 사이에 순환성이 있는지에 대한 검사를 하고, 이들이 의미상 일관되게 기술되었는지에 대한 평가를 하였다. 만약 추론엔진이 개발한 온톨로지에서 논리적으로 모순된 부분을 발견한 경우에는, 추론엔진은 이 부분에 대한 클래스나 정의된 개념을 반환한다. 본 논문에서는 이와 같은 방법을 통해 구문에러, 순환성 및 일관성을 검사하고, 오류가 없음을 확인하였다(그림 10).

5.2 완성도 평가

온톨로지의 완성도는 개념 정의에 대한 불충분성(incompleteness)과 지식 생략(knowledge omission)에 대한 검사로 확인한다[17,18]. 또한, 새로운 용어(term)

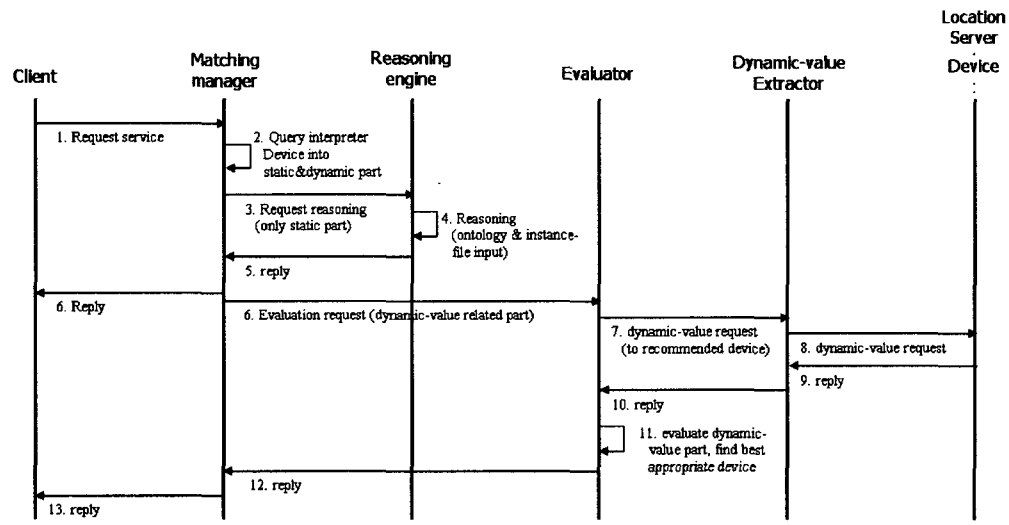

그림 9 서비스 요청 및 응답과정

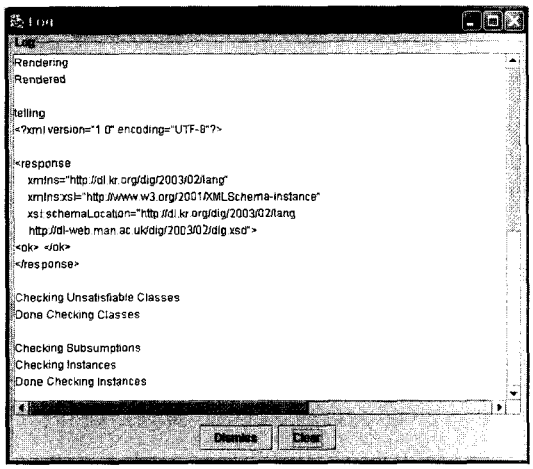

그림 10 OilEd를 사용하여 구문에러 및 일관성 검사

가 정의되더라도 기존의 온톨로지를 수정하지 않고 수용할 수 있는 확장성을 갖는지에 대한 평가를 통해 완성도를 체크한다.

본 논문에서는 개발한 온톨로지의 완성도를 평가하기 위하여 여러 시나리오 상에서 사용자가 다양한 서비스 요청 메시지를 생성하고, 온톨로지와 룩업서비스의 추론 엔진과 평가자를 통해 결과를 리턴 받는 것을 실험하였다. 실험 시나리오는 다음과 같다.

i. 김씨는 매일 아침 시끄러운 알람소리보다 오디오에서 흘러나오는 음악을 들으며 기상한다. 회사에 나가기에 앞서 그는 자신의 하루 스케줄을 확인하고 싶어서 스케줄 서비스를 요청하면, 해당 스케줄이 그와 가장 가까운 화면에 디스플레이 된다. 김씨가 회사에 나서면서 외출 자동모드의 서비스를 요청하면 모든 전등과 난방 장치가 자동으로 꺼지고, 문 잠금장치가 작동하여 창문과 현관문이 자동으로 잠긴다.

ii. 김씨가 하루일과를 마치고 집으로 돌아와서 현관에 들어서면, 센서가 김씨를 자동으로 감지하여 실내 자동모드 서비스를 요청한다. 서비스가 요청되면, 거실의 불이 켜지고, 난방 장치가 디폴트 값에 맞춰져서 작동된다. 또한, 자동응답기가 작동하여 새로운 메시지를 재생한다.

iii. 김씨는 잠들기 전에 영화를 보고 싶어서 DVD플레이어를 켜면, TV가 자동으로 켜지고 전등이 약한 불빛으로 바뀐다. 또한, 스피커의 볼륨이 켜지고 영화가 재생된다.

위의 시나리오는 사용자가 원하는 서비스나 디바이스의 스테틱/다이나믹 애트리뷰트를 이용하여 서비스 요청 메시지를 보내는 것이 가능하다. 아래의 표는 위의 시나리오에 해당하는 서비스 요청메시지와 이에 대한 결과값을 정리한 것이다.

위의 실험에서 볼 수 있듯이 우리가 개발한 시스템은 가능한 모든 타입의 쿼리 메시지를 처리하여 적절한 서비스를 검색한다. 사용자는 단일 서비스로 이루어진 쿼리메시지를 보내는 것뿐만 아니라, 여러 서비스가 동시에 만족되는 서비스를 요청할 수도 있다. 예를 들어, ②, ⑥, ⑦, ⑧의 쿼리 메시지는 사용자가 원하는 단일 서비스를 가지고 있는 실제 디바이스를 요청하고, ①의 경우는 사운드 서비스와 클락 서비스 두개를 동시에 만족하는 서비스를 찾는다. 이 경우는 사운드 서비스와 클락 서비스를 동시에 만족하는 서비스는 알람 서비스라는 것이 이미 온톨로지에 정의되어 있기 때문에, (concept-instances(and *SoundService ClockService*))의 쿼리메시지를 보내는 것도 가능하다. 다음으로 사용자는 단일 서비스와 원하는 디바이스의 특성을 제한하여 쿼리메시지를 보낼 수 있다. 예를 들어 ③, ⑩, ⑪의 경우는 사용자가 원하는 서비스와 요구되는 디바이스의 스테틱 애트리뷰트 또는 다이나믹 애트리뷰트를 기술하여 쿼리메시지를 보낸 것이다. 뿐만 아니라, ⑤, ⑨의 경우와 같이 라지 스크린과 같은 스테틱 애트리뷰트와 '같은 방'

Query Message	Result
① (concept-instances (and *SoundService ClockService*))	**Awia_MiniCombo**
② (concept-instances *SchedulerService*)	**KB_Scheduler**
③ (concept-instances (and *DisplayService CloestThing*))	**LGPlatronTV**
④ (concept-instances (or (or *LockService HeatingService*) *LightService*))	**SCDoorLock, SCWindowLock, KDBoiler, KDHeater Lamp_A0, Lamp_A1Lamp_A2, SamjungLamp**
⑤ (concept-instances (and (and *LightService BrightThing*) *LocatedinLivingRoom*))	**Lamp_A0, Lamp_A1Lamp_A2**
⑥ (concept-instances *BoilerHeatingService*)	**KDBoiler**
⑦ (concept-instances *AnsweringService*)	**BTAnsweringMachine**
⑧ (concept-instances *DVDPlayerService*)	**SSDVDPlayer**
⑨ (concept-instances (and (and *DisplayService LargeScreen*) *SameRoom*))	**SSPiPTV**
⑩ (concept-instances (and (and *LightService DimThing*) *SameRoom*))	**SamjungLamp**
⑪ (concept-instances (and (and *SoundService BigSoundThing*)*SameRoom*))	**AMPSpeaker**

Possible Service Request Format	Explanation
service	단일 프리미티브 서비스를 제공하는 디바이스 요청
service and service	두 개 이상의 프리미티브 서비스를 동시에 제공하는 디바이스 요청
service or service	A 또는 B 프리미티브 서비스를 제공하는 디바이스 요청
service and static device attribute and/or dynamic device attribute	하나의 서비스를 제공하면서 동시에 특정 애트리뷰트를 만족시키는 디바이스 요청

이라는 위치에 관련된 다이나믹 애트리뷰트를 함께 제한하여 쿼리메시지를 보낼 수 있다. 마지막으로, 사용자는 ④와 같이 여러 서비스를 동시에 요청하는 것이 가능하다.

실험 결과에서 알 수 있듯이 서비스 탐색 관리자는 단순한 구문 매칭을 통한 서비스 검색이 아니라, 시멘틱 서비스 탐색 관리자를 이용하여, 서비스와 디바이스가 갖는 시멘틱을 고려하여 서비스를 검색한다. 또한, 위치와 같이 동적으로 변하는 디바이스의 특성도 반영하여 서비스가 검색되도록 한다.

우리가 개발한 온톨로지는 일반 중산층 가정환경을 가정하여 온톨로지를 개발하였기 때문에, 현재까지 해당 도메인에 있는 인스턴스들은 정의되어 있는 클래스와 속성을 이용하여 모두 표현이 가능하다. 만약 새로운 디바이스나 서비스, 용어 등이 정의 되었을 때는 본 논문에서 정의한 각 클래스의 계층분류나 하위 클래스로 정의된 기타 클래스를 이용하여 이를 표현하고, 표현된 클래스나 프로퍼티를 이용하여 인스턴스를 정의한다. 따라서 본 논문에서 개발한 온톨로지는 퍼베이시브 홈 네트워크 환경이라는 도메인의 특성을 반영하여 개념의 불충분성 없이 디바이스나 서비스를 기술할 수 있도록 하고, 새로운 용어가 정의되었을 때 기존의 온톨로지의 수정하지 않고 새로운 용어를 정의할 수 있는 확장성을 갖는다.

6. 결론 및 향후과제

본 논문에서는 퍼베이시브 홈 네트워크 환경을 위한 온톨로지를 설계 및 개발하고, 개발된 온톨로지를 이용하여 시멘틱 서비스 검색이 가능하게 하였다. 개발된 온톨로지의 주요 목적은 가정환경내의 디바이스와 서비스를 효과적으로 표현하는 것이다. 이를 위해 우리는 디바이스, 서비스, 프리미티브 서비스, 애트리뷰트, 상태변수, 컨트롤 인터페이스 클래스를 정의하였고, 이들 간의 관계를 기술하였다. 특히, 프리미티브 서비스 클래스를 통해 디바이스 클래스와 서비스 클래스간의 관계를 효과적으로 나타내고, 디바이스가 갖는 시멘틱을 보다 풍부하게 표현할 수 있었다.

개발된 온톨로지는 실제 서비스 탐색 시스템을 통해 이용된다. 우리는 이를 위해 기존의 Jini 룩업서비스를 확장하여 서비스 탐색 시스템을 개발하였다. 확장된 룩업서비스는 추론엔진을 포함하고, 컨텍스트 정보에 대한 평가를 위해 평가자를 포함한다. 우리는 개발한 시스템과 온톨로지의 완성도를 평가하기 위하여, 다양한 시나리오 상에서 여러 타입의 서비스 요청메세지를 보내고, 이를 해석하여 적절한 서비스를 검색할 수 있는지에 대한 평가를 하였다. 또한, 구문에러 등의 문제가 있는지에 대한 기술적 평가를 툴을 이용하여 검사하였다. 그 결과, 개발한 온톨로지가 구문에러나 일관성 오류 등이 없음을 확인하였고, 퍼베이시브 홈 네트워크 환경의 디바이스와 서비스를 효과적으로 빠짐없이 기술하였음을 확인하였다.

참 고 문 헌

[1] M. Satyanarayanan, "Integrated pervasive computing environments," *IEEE Pervasive Computing* Vol. 1, Issue 2, pp. 2 -3, April-June 2002.

[2] P. Borst, H. Akkermans and J. Top "Engineering Ontologies," *International Journal of Human-Computer Studies*, pp. 365-406, 1997.

[3] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intelligent Systems*, Vol. 16, No. 2, pp. 72-79, 2001.

[4] D. Fensel and M.A. Musen, "The Semantic Web: A Brain for humankind," *IEEE Intelligent Systems*, March-April 2001.

[5] D. Chakraborty, F. Perich, S. Avancha and A. Joshi, "DReggie: Semantic Service Discovery for M-Commerce Applications," *Symposium on Reliable Distributed Systems*, 2001.

[6] S. Avancha, T. Finin, and A. Joshi, "Enhanced service discovery in bluetooth," In *IEEE Computer*, June 2002.

[7] A. Ranganathan, R.E. McGrath, R.H. Campbell, and M.D. Mickunas, "Ontologies in a Pervasive Computing Environment," In *Workshop on Ontologies and Distributed Systems (part of the 18'th International Joint Conference on Artificial Intelligence (IJCAI 2003))*, Acapulco, Mexico, 9 August 2003.

[8] H. Chen, T. Finin, and A. Joshi. "An Ontology for Context-Aware Pervasive Computing Environments," *Ontologies for Distributed Systems*, Knowledge Engineering Review, November 2003.

[9] N.F. Noy, and D.L. McGuinness. "Ontology Development 101: A Guide to Creating Your First

Ontology," Knowledge Systems Laboratory, March 2001.
[10] OilEd, http://oiled.man.ac.uk
[11] Jini, http://www.jini.org
[12] UPnP, http://www.upnp.org
[13] E.Guttman, C.Perkins, J.Veizades, and M.Day. "Service Location Protocol, Version 2," http://www.rfceditor.org/rfc/rfc2608.txt, 1999.
[14] UDDI, http://www.uddi.org
[15] Object Management Group, "CORBA services: Common Object Services Specification," Object Management Group, 1999.
[16] Sun Microsystems, "Java Remote Method Invocation," http://java.sun.com/products/jdk/rmi/
[17] A. Gomez-Perez, and M.C. Suarez-Figueroa, "Results of Taxonomic Evaluation of RDF(S) and DAML+OIL ontologies using RDF(S) and DAML+OIL Validation Tools and Ontology Platforms import services," *EON 2003 2nd International Workshop on Evaluation of Ontology-based Tools*, 2003.
[18] A. Gomez-Perez, "Some ideas and examples to evaluate ontologies," In *Proceedings of the Eleventh Conference on Artificial Intelligence Applications, IEEE Computer Society Press*, 1995.
[19] RACER, http://www.sts.tu-harburg.de/~r.f.moeller/racer
[20] FaCT, http://www.cs.man.ac.uk/FaCT

조 미 영

2002년 2월 성균관대학교 사범대학 컴퓨터교육과 학사. 2004년 2월 한국정보통신대학교 공학석사. 2004년 1월~현재 삼성전자 정보통신총괄 무선사업부 연구원

강 세 훈

1995년 2월 고려대학교 전산과학 학사 1995년 2월~1998년 3월 ㈜LG반도체 Design Technology 연구소 연구원 1999년 8월 한국정보통신대학교 공학석사. 1999년 9월~현재 한국정보통신대학교 박사과정

이 영 희

1976년 2월 서울대학교 공과대학 공업교육학과 공학사. 1980년 2월 서울대학교 공과대학 공업교육학과 공학석사. 1984년 6월 프랑스 UTC 전산학 박사. 1984년 8월~1997년 12월 ETRI, 정보통신표준연구센터 센터장. 1986년 7월~1987년 11월 IBM T.J.Watson 연구소 초빙과학자. 1998년 1월~현재 ICU 교수(공학부장, SOTI 연구소장 역임)