논문 2008-45CI-3-11

유비쿼터스 환경에서의 상황인지 기반 디바이스 협업 시스템

(A Context-Based Device Collaboration System in Ubiquitous Environments)

박 원 익*, 박 종 현**, 김 영 국***, 강 지 훈***

(Won-Ik Park, Jong-Hyun Park, Young-Kuk Kim, and Ji-Hoon Kang)

요 약

유비쿼터스 환경은 보이지 않는 수많은 장치들과 소프트웨어들이 서로 연결되어 각각의 사용자들에게 편리한 서비스를 제공한다. 이러한 서비스를 제공 받기 위해서는 사용자와 서비스간의 매개체 역할을 하는 모바일 단말기가 필요하다. 하지만 리소스가 제한적인 모바일 단말기의 특성상 다양한 기능을 이용할 수는 없다. 따라서 본 논문에서는 상황인지 기반의 실시간 디바이스 협업 시스템을 개발하여 주변의 다양한 디바이스 리소스를 실시간으로 공유함으로써 모바일 단말기의 제한적인 리소스 문제를 해결하는 상황인지 기반의 디바이스 협업 시스템을 제안한다. 제안된 시스템의 특징은 변화하는 사용자의 선호도 및 리소스 정보를 동적으로 재구성하여 상황에 맞는 맞춤형 디바이스 리소스의 공유가 가능하다는 점이다.

Abstract

In ubiquitous environments, invisible devices and software are connected to one another to provide convenient services to users. In order to provide such services, we must have mobile devices that connect users and services. However, the types of available services have thus far been limited due to the limited resources of mobile devices. This paper proposes a solution to the resource limitation problem of mobile devices by presenting a context-based collaboration system that allows mobile devices to share various nearby resources. Our system has a feature to enable personalized resource sharing by dynamically re-configuring user's preference and resource information.

Keywords : Context-based Collaboration System, Resource Sharing, Ubiquitous Computing, Ontology

Ⅰ. 서 론

최근 PDA, 스마트폰, 인터넷 단말기, 인터넷 TV. 스마트태그 등의 다양한 정보기기와 무선통신 및 센서 기술이 빠르게 보급, 발전됨에 따라 언제 어디서나 누구든지 각종 단말기와 장치들을 통해서 광대역 네트워크에 접속하여 서비스를 받을 수 있는 환경 즉, 유비쿼터스 환경이 현실화되고 있다[1~2].

이러한 유비쿼터스 환경의 특징은 다음과 같다[3~5]. 첫째, 회의실, 건물 등에 사용자의 위치를 파악할 수 있는 기반시설을 설치하여, 사용자 위치에 따른 효과적인 서비스를 제공할 수 있다. 둘째로 사용자는 유비쿼터스 컴퓨팅 기술을 사용하고 있다는 것을 느끼지 못하며, 셋째로 상황을 인지하여 그에 따라 다른 서비스를 제공한다. 마지막 네번째는 유비쿼터스 환경이 구축된 곳과 그렇지 않은 곳과의 서비스 차이를 사용자가 최소로 느끼게 해야 한다는 점이다. 이러한 유비쿼터스 환경을 본 논문에서는 USS(Ubiquitous Smart Space)라 정의한다. USS에서 사용자와 서비스간의 매개체 역할을 하는 모바일 단말기의 역할은 크다. 하지만 모바일 단말기의 제한적인 리소스만으로 제공하는 서비스를 원활히

* 학생회원, *** 정회원, 충남대학교 컴퓨터공학과
(Department of Computer Engineering, Chungnam National University)
** 정회원, 충남대학교 소프트웨어연구소
(Software Research Center, Chungnam National University)
※ 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅 및네트워크원천기반기술사업의 08B3-O1-30S 과제로 지원된 것임.
접수일자: 2008년4월26일, 수정완료일: 2008년5월6일

수행하기에는 한계가 있다. 따라서 본 논문에서는 제한적인 리소스 문제를 해결하기 위해서 주변의 다양한 디바이스 리소스를 실시간으로 공유하는 디바이스 협업 시스템을 제안한다.

예를 들면, 오전에 A라는 사람이 작은 모바일 디바이스로 동영상을 보면서 이동 중이다. 그러던 중에 A가 USS에 도착한다면 디바이스 협업 시스템은 A의 선호 정보 및 위치와 시간 같은 상황 정보를 고려하여 좀 더 편리한 디스플레이 장치를 추천, 공유 할 수 있게 해준다. 즉, 사용자 A는 모바일 단말기의 작은 화면 보다 선호하는 주변의 디스플레이 장치를 통해 끊김 없이 동영상을 볼 수가 있다. 이처럼 모바일 단말의 제한적인 리소스 문제는 주변의 디바이스 협업을 통해 보완할 수 있다.

모바일 단말기를 이용하여 위와 같은 자동적인 서비스를 구성하기 위해서는 서비스 요청 분석, 서비스에 해당하는 사용 가능한 디바이스 검색, 추론엔진, 사용자의 선호 정보, 사용자의 위치 및 현재 시간 등을 알아야 한다. 본 논문에서는 이들은 명료하게 정의된 형식과 의미를 제공하는 온톨로지로 표준에 맞추어 기술하고 필요한 지식들을 규칙으로서 정의하였다. 그리고, 제안하는 디바이스 협업 시스템의 가능성을 검증하기 위해 예제 시나리오를 구성하였다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 관련연구로서 IBM의 Celadon 프로젝트, UMBC(University of Maryland, Baltimore Country)의 CorBra 프로젝트, University of Illinois, Urbana Champaign 의 GAIA 프로젝트에 대해 알아보고 Ⅲ장에서는 본 논문에서 제안하는 디바이스 협업 시스템 구조를 설명하며, Ⅳ장에서는 본 논문에서 구성한 온톨로지에 대해서 설명한다. Ⅴ장에서는 시나리오를 통한 상황인지 기반의 디바이스 협업을 통해 동적인 서비스를 구성하는 예제 시나리오를 제시한다. 마지막으로 Ⅵ장에서는 결론 및 향후 과제에 대하여 기술한다.

Ⅱ. 관련 연구

1. Celadon Project

Celadon 프로젝트는 상황 인지 기반의 지능형 유비쿼터스 협업 시스템을 구축하기 위해 IBM에서 2004년부터 추진하는 프로젝트로써 사용자 주변에 다양한 서비스, 디바이스가 존재하는 유비쿼터스 환경에서 지적 객체간의 지능적인 협업 및 상황인식 서비스를 제공해 주는 브로커 중심의 미들웨어 인프라스트럭처 구축을 주된 목표로 하여 개발하고 있다[6~7]. 유비쿼터스 환경에서 수많은 서로 다른 기종의 무선 단말기를 주변 장비와 상호 무선통신이 가능하도록 특정 지역(Zone) 내에서 상호 협업(Collaboration)을 할 수 있도록 하자는 것이다. 운영시스템(OS)과 플랫폼이 다른 여러 가지 기기나 장비들을 서로 연동해 사용자 맞춤형 서비스를 제공한다. 쉽게 표현하면 어떤 기기로든, 어떤 장소에서든, 어떤 서비스라도 받을 수 있도록 하겠다는 것이다. 이를 위해서 Celadon은 동적 탐색 기술, 상황인식 기술, 디바이스 간 통신 기술, 디바이스 간 협업 기술과 같은 작업을 수행한다. 이를 위해 지능적인 협업 및 상황인식 서비스를 제공해 주는 브로커 중심의 미들웨어 인프라스트럭처를 구축해야한다.

2. CorBra (Context Broker Architecture)

UMBC(University of Maryland, Baltimore Country)에서 개발된 CoBrA는 유비쿼터스 환경에서 상황 정보의 생성, 관리, 배포를 위해 공간상에 존재하는 모든 컴퓨팅 엔티티들을 위한 상황 공유모델을 관리하는 브로커 에이전트이다[8~9]. CoBrA의 특성은 상황을 RDF와 OWL을 이용하여 정형화된 틀에 기반을 둔 온톨로지로 구성함으로써 지식에 대한 공유를 가능하게 했으며, 이를 지식 기반으로 구성하여 상황에 대한 기본적인 추론이 가능하도록 하였다. 센서를 이용한 정형화된 사용자 행동이나 공간 개념의 정보, 디바이스 및 특정 상황(회의, 문서, 시간)에 대한 것을 온톨로지로 정의하여 특정 상황에 특화된 서비스 및 정보를 관리하였다. 중앙화된 상황 모델로 구성하여 상황에 대한 일관적인 관리의 효율성을 높였으며, 논리적 불일치에 대한 수정이 가능하도록 구현하였다. 하지만 중앙 집중식의 데이터 처리는 이동형 환경에서 적용하기 취약하다. 또한, 상황인식 컴퓨팅 환경의 모든 측면을 다루지 않는 단점이 있다.

3. GAIA: A Middleware Infrastructure to Enable Active Space

University of Illinois, Urbana Champaign에서 연구된 GAIA는 물리적인 공간과 그 공간 내에 존재하는 리소스들을 프로그래밍이 가능한 전자 공간으로 변환하는 기능을 수행한다[10]. 유비쿼터스 컴퓨팅 환경에서의 여러 문제를 해결하기 위해 온톨로지가 효과적으로 적용 가능함을 보여주고 있으며 분산된 객체 간의 연동을 위해 안정적 구조를 제공할 수 있는 CoBrA 미들웨어의

적용을 기반으로 온톨로지에 대한 추론 기능을 구현하였다. 서비스를 제공하는 컴퓨팅 환경을 Active Space라 정의하고 그 Active Space는 사용자 중심 서비스 제공 엔터티를 포함하고 있는 물리적인 경계가 있는 공간으로 보고 있다. 사용자의 요구에 맞는 서비스를 제공하기 위한 여러 가지 기술을 사용하고 있다. High level-abstraction 기법, Programming model, Extended MVC model, Service script(ACD)를 사용했다. GAIA는 Application Framework과 Component Management Core 사이에 있는 부분에서 상황인식 서비스에 관련된 일을 수행한다.

이처럼 GAIA는 상황인식 서비스 구조로 응용이 다양한 상황정보를 얻고 추론할 수 있게 해주지만 중앙집중식 데이터베이스를 활용하여 구축된 프레임웍으로 이동형 환경에서 적용하기에는 취약하다.

위에서 언급한 컨텍스트 어웨어 시스템들은 이동 에이전트 기반이거나 클라이언트/서버 기반이다. 하지만 본 논문에서 제안하는 상황인지 기반 디바이스 협업 시스템은 기본적으로 Peer to Peer 형식으로 작동한다. 따라서 별도의 서버나 에이전트 환경 구축 없이 서비스가 가능하며 중앙 집중적인 데이터베이스를 활용하지 않아 이동형 환경에 적합하다. 또한, 동적으로 변화하는 사용자 관련 정보 및 가용한 리소스 정보를 고려하여 보다 정확한 디바이스 협업 환경을 제공하는 장점이 있다.

Ⅲ. 상황인지 기반 디바이스 협업 시스템 구조

상황인지 기반 디바이스 협업 시스템은 그림 1과 같이 Service Broker, Situation-Aware Service Engine, Resource Composer, Resource Discoverer로 구성된다.

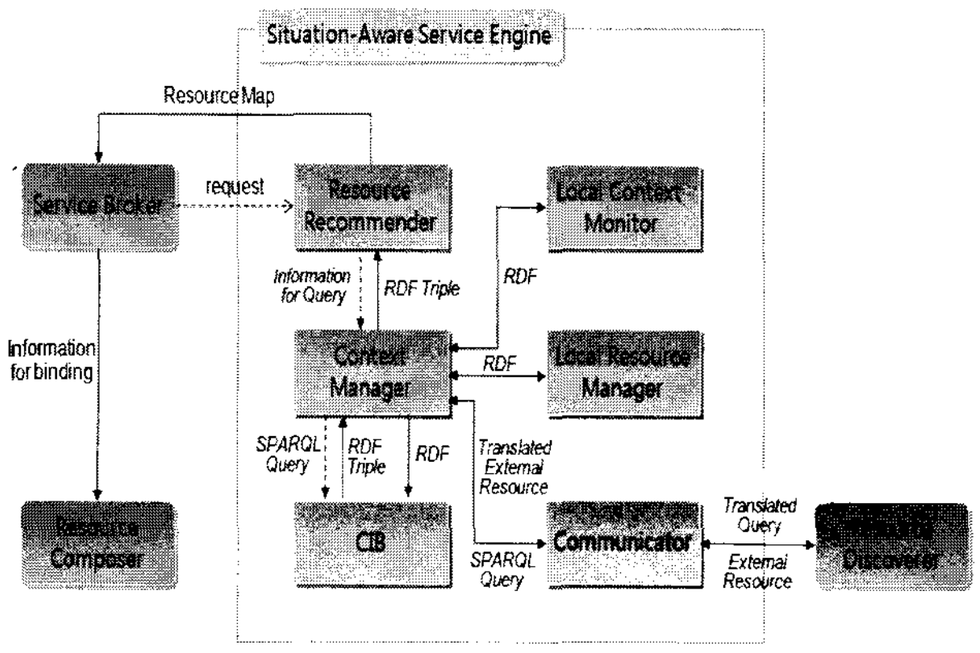


그림 1. 상황인지 기반 디바이스 협업 시스템 구조
Fig. 1. Context-Based Device Collaboration System Architecture.

Service Broker는 사용자의 서비스 요청 및 결과를 처리하는 역할을 하고 Situation-Aware Service Engine은 요청한 서비스명과 관련된 서비스 디바이스들을 분석하여 디바이스 및 상황 정보를 저장하고 있는 CIB (Context Information Base)에 요청하여 얻은 정보를 이용하여 상황 및 사용자의 선호도를 고려한 디바이스를 추천 해주는 역할을 한다. Resource Discoverer는 이용 가능한 디바이스를 탐색한다. 마지막으로, Resource Composer는 실제 추천된 이용 가능한 디바이스를 바인딩하기 위해 필요한 정보를 수집하여 연결해주는 역할을 한다.

본 논문에서는 Situation-Aware Service Engine을 중심으로 설명한다.

1. Situation-Aware Service Engine

Situation-Aware Service Engine의 역할은 USS를 인지하고, 사용자의 모바일 단말기에서 요청한 서비스를 제공하기 위해 디바이스 리소스를 전문가 시스템을 사용해서 효율적으로 찾는 것이다. 또한 본 연구에서 제안하고 있는 Situation-Aware Service Engine은 다양한 USS에 적합하도록 각 모듈의 확장을 고려하여 설계한다.

그림 2는 Situation-Aware Service Engine의 구성도이며 이를 기반으로 하는 동작 시나리오의 예는 다음 단계를 따른다.

STEP 1 :

Service Broker로부터 서비스가 요청되기 전에 Local Context Monitor와 Local Resource Manager에서 수집

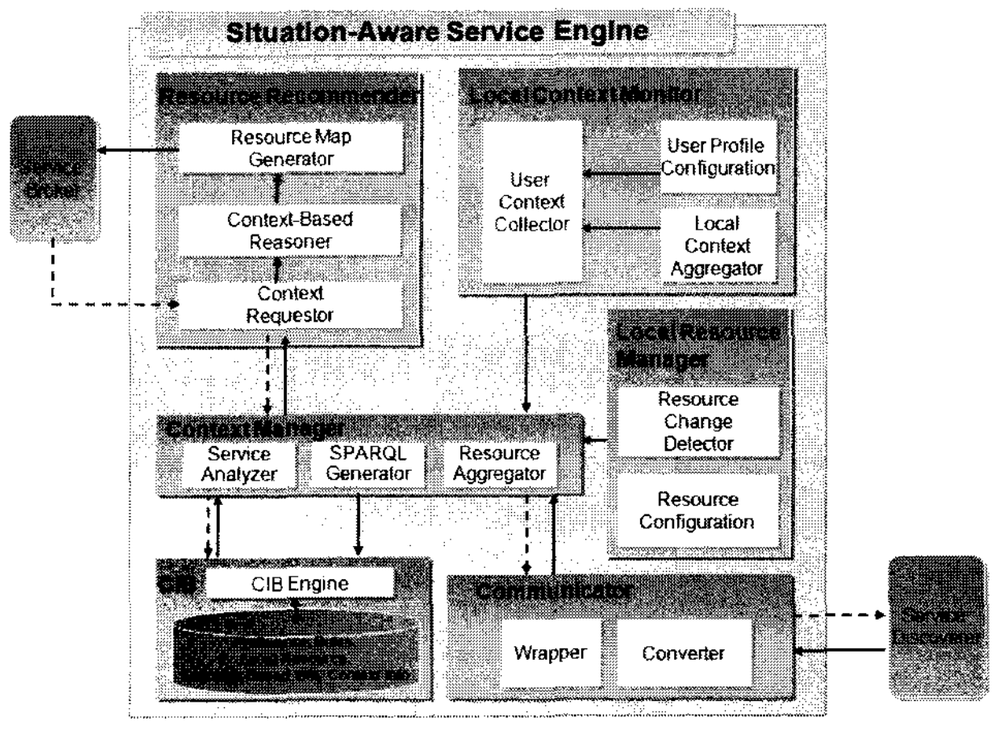


그림 2. Situation-Aware Service Engine 구성도
Fig. 2. Architecture of the Situation-Aware Service Engine.

한 추천에 필요한 정보들을 CIB에 RDF형식으로 저장한다.

STEP 2 :

Service Broker로부터 서비스명(예를 들면, "문서편집")이 전달된다.

STEP 3 :

Resource Recommender의 Context Requestor 모듈은 Ontology Schema로부터 서비스 요청을 맵핑하기 위해 필요한 정보를 Context Manager에게 전달한다.

STEP 4 :

Context Manager의 SPARQL Generator 모듈은 CIB에 저장된 RDF를 검색하기 위해 SPARQL 질의를 생성한다.

STEP 5 :

Resource Recommender의 Context Requestor 모듈은 Context Manager를 통해 질의 결과를 받는다. 그리고 질의 결과(예를 들어, 모니터에 대한 RDF Triple)는 Context-Based Reasoner 모듈의 입력(Fact)으로 주어진다. 다른 입력 값인 룰은 CIB에 저장된 값을 이용한다.

STEP 6 :

Resource Recommender의 Context-Based Reasoner 모듈은 정의된 룰과 Context Requestor로부터 넘겨받은 정보를 이용하여 상황에 맞는 디바이스 리스트를 추론한다. 추론된 결과인 상황에 따른 사용자 맞춤형 디바이스 정보를 Resource Map Generator 모듈에 전달한다. 이와 함께 추론 결과는 User Context Monitor를 통해서 이전의 히스토리를 갱신한다.

STEP 7 :

Resource Map Generator 모듈은 추론 결과인 디바이스 정보로부터 디바이스의 ID 및 IP 주소를 추출한다.

STEP 8 :

생성된 Resource Map은 Resource Broker 모듈에 전달된다.

가. Resource Recommender

이 모듈의 역할은 Service Broker로부터 입력된 서비스 요청을 처리하기 위해 동작하는 모듈로서 궁극적으로 SPARQL 질의를 생성하기 위한 정보를 제공하고 질의의 결과인 RDF Triple를 얻어 와서 상황에 따른 맞춤형 디바이스 정보를 제공하는 일을 담당한다. 이를 위해서 UMO(Ubiquitous Mobile Object) 사용자의 서비스 요청에 해당하는 정보를 수집하기 위한 질의 정보를 Context Manager에게 제공하고 질의 결과인 RDF Triple정보를 Context-Based Reasoner에게 넘기는 Context Requestor 모듈과 RDF Triple 정보와 룰 및 사용자의 선호도 정보를 이용하여 디바이스를 추천 해주는 Context-Based Reasoner 모듈이 있다. 추천된 디바이스에 대해서 Resource Map Generator는 바인딩 시 필요한 디바이스 ID와 IP 주소 정보로 가공한 리소스 맵을 Service Broker에게 보낸다. 그림 3은 Resource Recommender의 모듈 구성도이다.

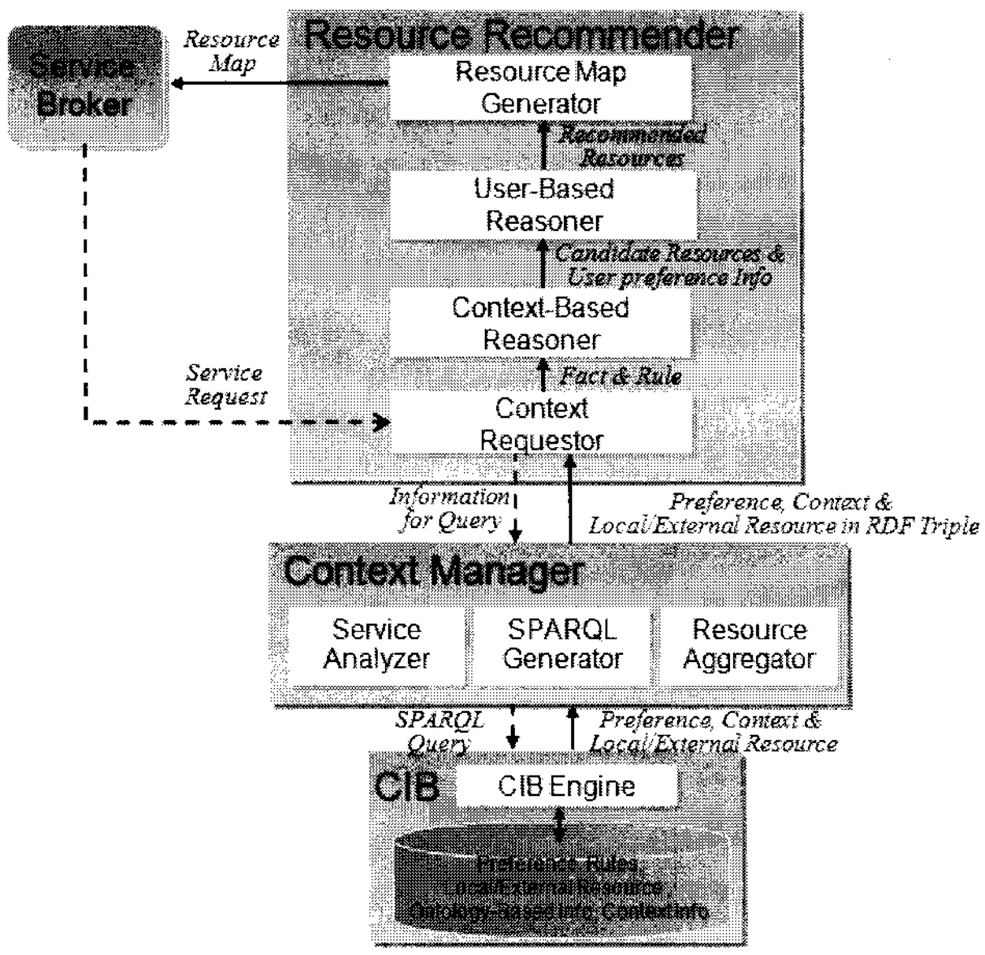


그림 3. Resource Recommender 구성도
Fig. 3. Architecture of the Resource Recommender.

(1) Context Requestor

Context Requestor 모듈은 사용자의 서비스 요청을 입력 받아 질의를 위한 정보를 Context Manager에게 전달하고 질의 결과인 RDF Triple 정보를 리턴 받는 인터페이스를 정의하는 모듈이다. 또한 RDF Triple과 정의된 룰을 조합하여 Context Requestor 모듈에서 사용하는 룰 기반 추론 엔진인 JESS의 입력값 .jess 파일을 생성하는 역할을 담당한다.

(2) Context-Based Reasoner

Context-Based Reasoner 모듈에서 사용되는 JESS 엔진은 Sun Microsystems에서 제작한 룰 기반의 전문가 시스템 저작 도구이다[11]. JESS는 본래 CLIPS 전문가 시스템 저작 도구에서 유래되었고, Ontology와 Rule을 기반으로 추론한다. 본 논문에서는 JESS를 사용하여 정의한 Ontology와 Rule들을 입력으로 현재 모바일 단말기의 상황에 맞는 추론 결과를 제공한다. JESS는 Java 기반으로 구현되어 있으므로 응용에 독립적으로 동작한다는 특성을 가지며, 현재 7.01 Version이 사용되고 있다.

JESS의 입력은 JESS function, *.clp, *.jess, *.xml로 작성된 디바이스 정보(Fact)와 룰의 조합이다. 그리고 출력은 추론된 디바이스들의 리스트이다. 표 1은 JESS 엔진을 이용한 추론에 사용된 함수들이다.

룰 기반 추론은 조건이 제시되고 조건이 만족하는 경우 결과를 도출할 수 있도록 구성된다. 따라서 룰 기반 추론은 시간이나 위치와 같은 상황에 따른 디바이스 추론을 하기 위해서 각 상황에 맞는 다양한 룰을 기술해야 한다.

표 1. 추론에 사용된 JESS function
Table 1. JESS Functions Used for Inference.

Function	Description
deftemplate()	Default template Fact에 대한 Type정의
defrule()	Default Rule 사용자 룰을 기술
deffacts()	Default Facts()에 정의한 Instance를 기술
assert()	Fact를 추가
retract()	Fact를 제거

(3) Resource Map Generator

Resource Map Generator 모듈은 User-Based Reasoner로부터 추론된 상황을 고려한 사용자 맞춤형 디바이스 정보를 Service Broker에게 전달하는 역할을 한다. Service Broker는 Resource Composer에게 정보를 전달한다. Resource Composer는 전달받은 디바이스 정보를 실제로 사용할 수 있도록 하드웨어적인 구성을 실행하는 역할을 한다. 즉, Resource Map Generator는 추론된 디바이스 정보에 대한 하드웨어 구성을 Resource Composer에게 전달하는 역할을 한다.

생성된 리소스 맵은 Resource Composer에 전달된다. 이때 Resource Composer는 맵의 Device ID 정보와 IP 정보를 이용해서 해당 Device를 바인딩한다. 그림 4는

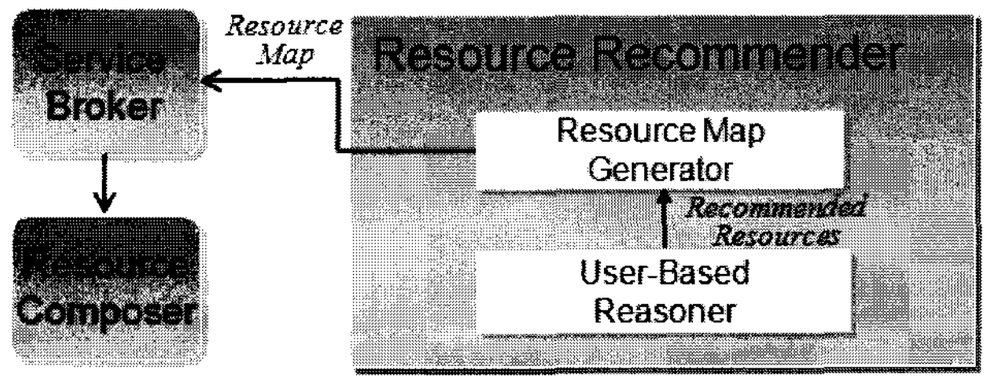


그림 4. Resource Map Generator 구성도
Fig. 4. Architecture of the Resource Map Generator.

Resource Map Generator 모듈과 상호작용하는 모듈간의 구성도를 나타낸다.

User-Based Reasoner로부터 받은 디바이스 정보는 리소스 맵을 구성하기 위한 최소의 집합으로 이루어져 있다. 다음은 User-Based Reasoner의 결과 예를 보여준다.

Monitor, Monitor01, 168.188.128.111
Monitor, Monitor03, 168.188.128.101
Mouse, Mouse01, 168.188.128.113
Keyboard, Keyboard03, 168.188.128.111

나. Local Context Monitor

사용자의 성향과 시간, 위치와 같은 환경 상황 정보는 사용자에게 디바이스를 추천할 때 중요한 정보이다. 본 논문에서는 시뮬레이션을 이용하여 동적으로 변화하는 가상의 사용자, 디바이스 및 환경 상황 정보를 Local Context Monitor의 User Profile Configuration 모듈과 Local Context Aggregator 모듈을 통해 수집한다. 수집된 정보를 분석하여 사용자의 선호도 관련 정보 및 환경 상황 정보를 RDF 형태로 변환하여 Context Manager를 통해 CIB에 저장한다. 이러한 과정은 사용

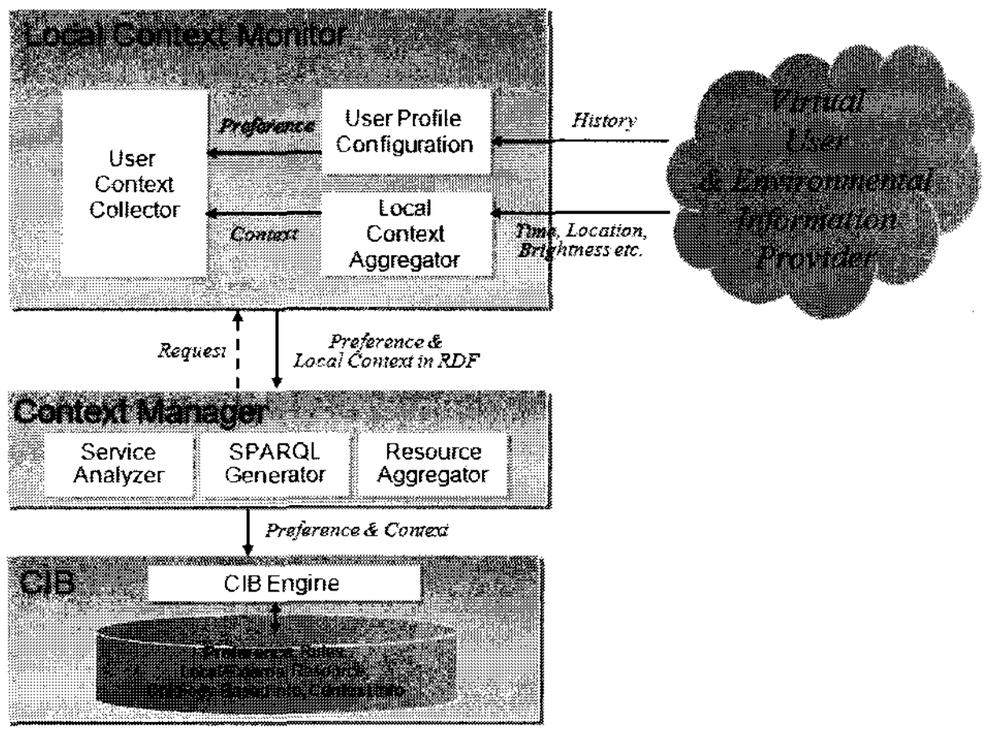


그림 5. Local Context Monitor 구성도
Fig. 5. Architecture of the Local Context Monitor.

자의 선호도 관련 정보 및 환경 상황 정보의 변화 시 재 수행된다.

본 논문에서는 다양한 사용자의 사용 어력이나 시간, 위치와 같은 환경 상황 정보를 가상으로 생산하여 동적으로 변화하는 환경을 구축한다.

(1) Virtual User Information

UMO 사용자가 과거에 사용한 디바이스 정보와 같은 사용 이력들을 기술한 정보를 생산한다.

(2) Virtual Environmental Information

시간이나 위치, 밝기, 온도와 같은 환경 상황 정보를 동적으로 기술한 정보를 생산한다.

다. Local Resource Manager

동적으로 변화하는 가용한 디바이스 정보를 자동 구성하는 역할을 하는 Local Context Monitor는 Resource Change Detector 모듈을 통해 추가 삭제된 디바이스에 대한 탐지를 한다. 탐지된 변경정보를 Resource Configuration 모듈은 RDF형식으로 변환하여 Context Manager를 통해 CIB에 업데이트한다.

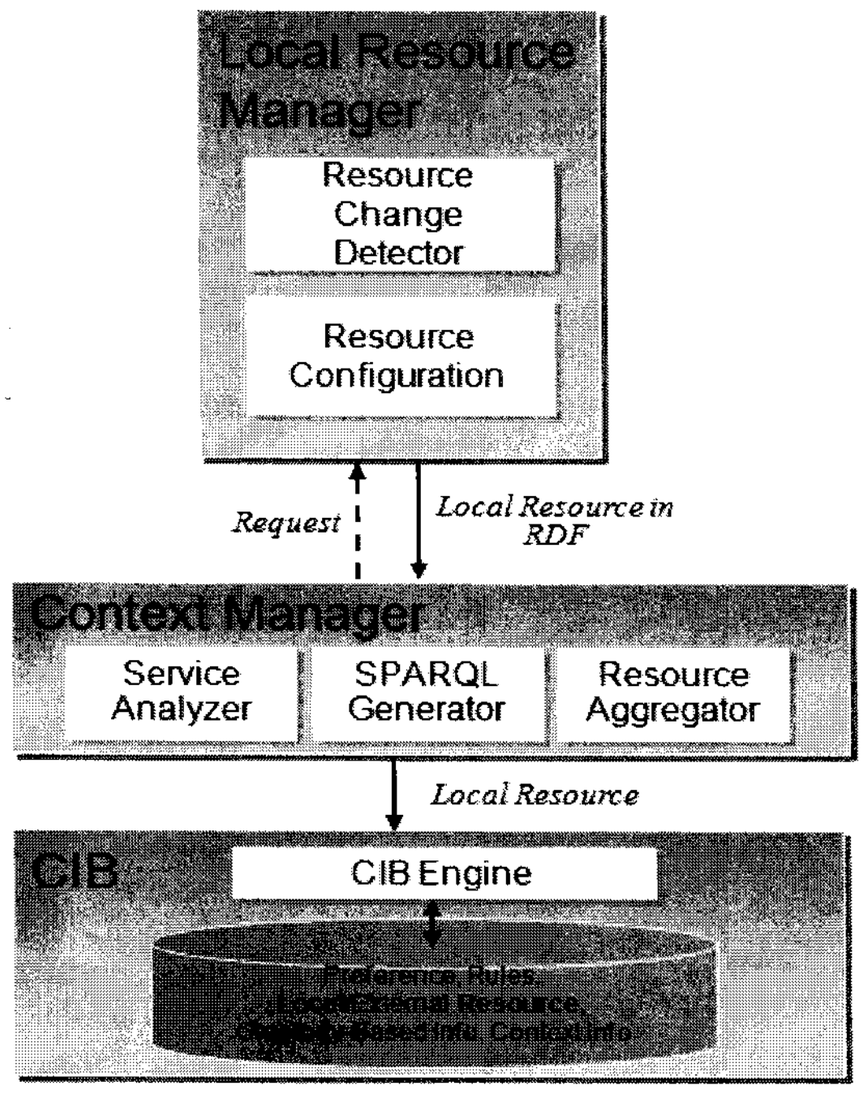


그림 6. Local Resource Manager 구성도
Fig. 6. Architecture of the Local Resource Manager.

라. Context Manager

Context Manager의 SPARQL Generator 모듈은 사용자가 요청한 Service Request 정보를 이용하여 CIB에 저장된 정보 검색 요청을 위한 SPARQL 질의를 생성하는 모듈이다. SPARQL은 Triple로 저장된 RDF를 검색하기 위한 질의 언어이다. 문서편집 작업을 위해 필요한 모니터 장치 리소스를 검색하기 위한 SPARQL 질의 예제는 아래와 같다.

SPARQL query 예제

```
PREFIX umo:<http://www.keti.re.kr/ubicom/umo#>
PREFIX ns:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  SELECT ?x
  WHERE {
    ?x ns:type umo:Monitor .
    ?x umo:Monitor_horizon_resolution ?y.Filter (?y >= \"1024\")
    ?x umo:Monitor_vertical_resolution ?z.Filter(?z >= \"768\")
}
```

생성된 질의는 CIB로 전달되며, 검색된 질의 결과는 RDF Triple 형태로 변경하여 Resource Rcommender로 전달된다.

SPARQL query 결과

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY
   rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY umo 'http://www.keti.re.kr/ubicom/umo#'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
  xmlns:umo="&umo;"xmlns:rdfs="&rdfs;">
<umo:LocalResource rdf:about="&umo;LocalResource_01"
  rdfs:label="LocalResource_01"/>
<umo:UMO rdf:about="&umo;UMO_01"
  umo:ID="KETI_UMO_ID_001"umo:annotation="KETIMobile
  device is..." rdfs:label="UMO_01">
  <umo:HasLocalDevice
  rdf:resource="&umo;LocalResource_01"/>
  <umo:communicationOf
  rdf:resource="&umo;Zigbee__001"/>
</umo:UMO>
<umo:UMPC rdf:about="&umo;UMO_Instance_2"
umo:UMPC_driver_position="e://driver/Umpc.dll"
   umo:UMPC_driver_size="10"umo:annotation="UMPC
   Driver..." rdfs:label="UMO_Instance_2"/>
<umo:Zigbee rdf:about="&umo;Zigbee__001"
rdfs:label="Zigbee__001"/>
<umo:Monitor rdf:about="&umo;monitor_01"
   umo:ID="KETI_WALLDISPLAY_0001"
   umo:Monitor_horizon_resolution="1024"
```

```
    umo:Monitor_is_LCD="true"
    umo:Monitor_size="32inch"
    umo:Monitor_vertica_resolution="768"
    umo:annotation="WallDisplay is.."
    umo:vender="KETI" rdfs:label="monitor_01">
    <umo:IsLocalResourceOf
      rdf:resource="&umo;LocalResource_01"/>
    <umo:hasDriver rdf:resource="&umo;UMO_Instance_2"/>
</umo:Monitor>
</rdf:RDF>
```

마. Communicator

Communicator는 외부 UMO 디바이스에 대한 가용한 리소스 정보 요청 질의 및 검색된 리소스 목록 결과를 이용하여 생성된 RDF 리소스 정보를 전달 받는 역할을 수행한다. SPARQL을 지원하지 못하는 경량 버전의 UMO 프레임워크와의 원활한 상호교환을 위해 Converter를 이용한다. Converter 모듈에서는 SPARQL 질의를 자체적으로 정의한 질의 포맷으로 변환하는 역할과 반대로 자체 쿼리 형태를 SPARQL 질의 형태로 변환하는 역할을 한다. 또한, SPARQL 질의에 대한 결과인 RDF 형태를 자체 정의한 토큰 타입의 데이터 형태로 변환시켜주는 역할과 정의한 토큰 타입의 데이터 형태에서 RDF 형태로 변환하는 역할을 수행한다. 자체 정의한 질의 Syntax 및 정의한 데이터 포맷은 다음과 같다.

- subquery ::= RETURN CONTEXT FROM
  context_type WHERE search_condition |
  subquery UNION subquery
- search_condition ::= property operator value
  |search_condition {AND | OR}
  search_condition | NOT(search_condition)
- context_type ::= literal
- operator ::= > | < | = | <> | >= | <=
- value ::= integer | real | literal | boolean
- integer ::= -?[0-9]+
- literal := '[a-zA-Z]+'
- boolean := TRUE | FALSE

자체 정의한 데이터 포맷은 기본적으로 RDF Triple에서 제공하는 정보의 내용을 모두 포함한다.

Context Type | *Context Value* | *Entity ID*

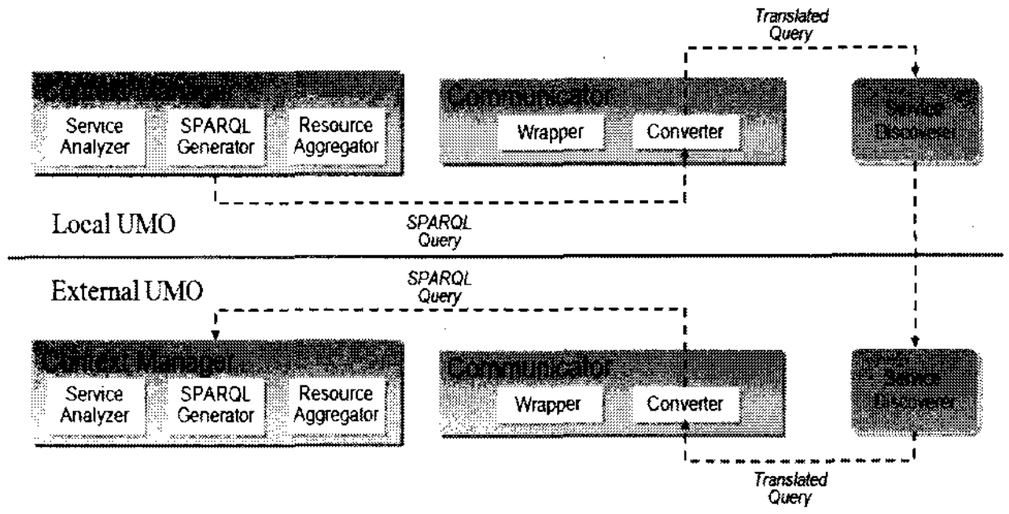


그림 7. 외부 리소스 정보 요청

Fig. 7. Request for External Resource Information.

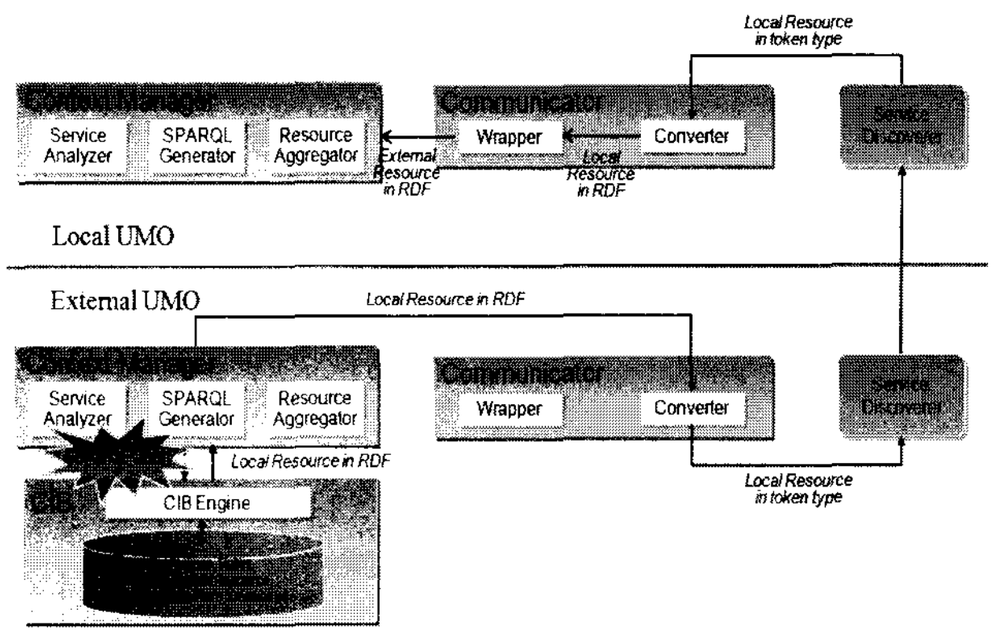


그림 8. 외부 리소스 정보

Fig. 8. External Resource Information.

그림 7은 로컬 UMO에서 외부 UMO 디바이스에 대한 가용한 리소스 정보를 요청하는 질의 시 흐름도이다.

그림 8은 Communicator의 또 다른 역할 중 하나인 그림 7과 같은 외부 리소스 정보 요청 질의에 대한 검색 결과인 RDF 리소스 정보를 Context Manager의 Resource Aggregator에 전달하는 과정을 보인다.

Communicator의 Wrapper 모듈은 Service Discoverer로부터 전달 받은 공유 리소스의 정보를 현재 시스템의 로컬 정보를 기반으로 재구성하는 작업을 수행한다. 예를 들어, 검색 요청된 외부 UMO의 리소스 정보는 외부 UMO 입장에서는 로컬 리소스 정보이다. 하지만 질의를 요청한 UMO 입장에서는 외부 리소스 정보가 된다.

Ⅳ. Ontology

온톨로지는 개념과 관계들로 구성된 사전으로 특정 도매인에 관련된 객체들을 계층적 구조로 표현하고 추가적으로 이를 확장할 수 있는 추론 규칙을 포함한다[12~13]. 온톨로지의 역할은 다양한 환경에서 서로 다른 단

어나 식별자를 사용할 경우에 이를 해결해 주는데 있다. 이러한 역할로 인하여 온톨로지는 웹 기반의 지식을 처리하거나 응용 프로그램 사이의 지식공유, 재사용 등을 가능하게 하는 아주 중요한 요소로 자리 잡고 있다. 본 논문에서는 이러한 온톨로지를 이용하여 지능공간과 리소스들의 개념과 그 관계를 정의하고 구성하여 서비스에 대해 적절한 디바이스를 찾는 방법을 제안한다.

1. 온톨로지 구성

본 논문에서 지능공간의 다양한 상황을 표현하기 위해 제안한 온톨로지는 크게 세 부분에 초점을 맞추어 설계한다.

첫 번째는 온톨로지 사용의 본래 목적인 지식 공유를 위하여 다양한 USS에서 범용으로 사용할 수 있는 Upper Ontology이다. 이는 우리의 온톨로지가 특정 공간에 의존적인 온톨로지가 아니고 다양한 USS에 유연하게 적용할 수 있도록 확장성을 부여한다. 아래의 그림 9는 본 논문에서 추론을 위해 필요한 Upper Ontology이다.

두 번째는 리소스 공유를 위한 다비이스 온톨로지이다. 리소스 공유를 위해서 DiscoveredResource, Local-Resource, VirtualResource, DeviceDriver, 그리고 Device의 서브 클래스들을 두어 보다 풍부한 리소스의 정보들을 표현한다. 이렇게 한번 분류된 디바이스들은 각각의 특성에 맞게 세분화하여 분류하였다. 리소스를 세분화하여 표현함으로써 리소스 공유의 관점에서 볼 때 상황에 부합하는 리소스를 찾고, 공유하기 위한 목적에 집약되는 온톨로지가 될 것이다. 그림 10은 세분화된 리소스를 보여주는 그림이다.

위와 같이 온톨로지의 구성을 세분화함으로써, CIB와의 상호작용에서 CIB를 통해 얻을 수 있는 지능공간의 상황과 조금 더 밀접한 관계를 가진 온톨로지를 표현할 수 있을 것으로 기대된다. 또한 CIB에서 얻어진

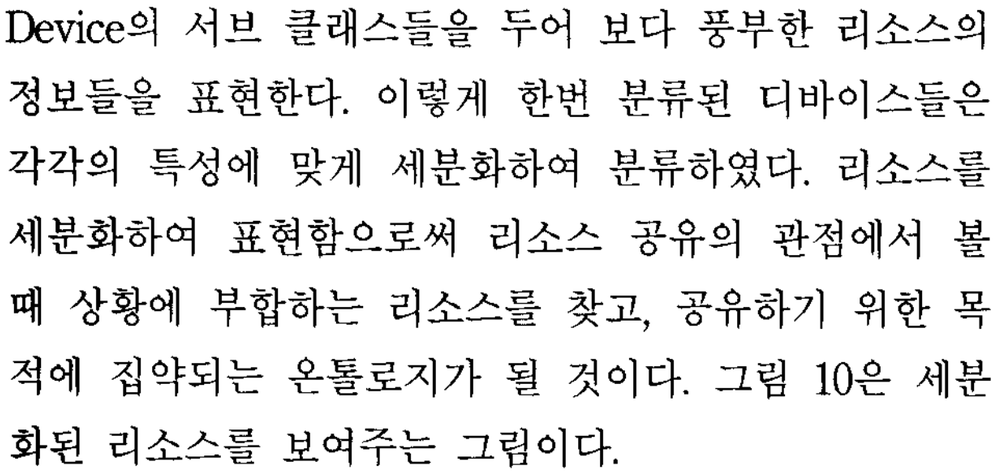


그림 9. 상위 온톨로지 클래스 다이어그램
Fig. 9. Upper Ontology Class Diagram.

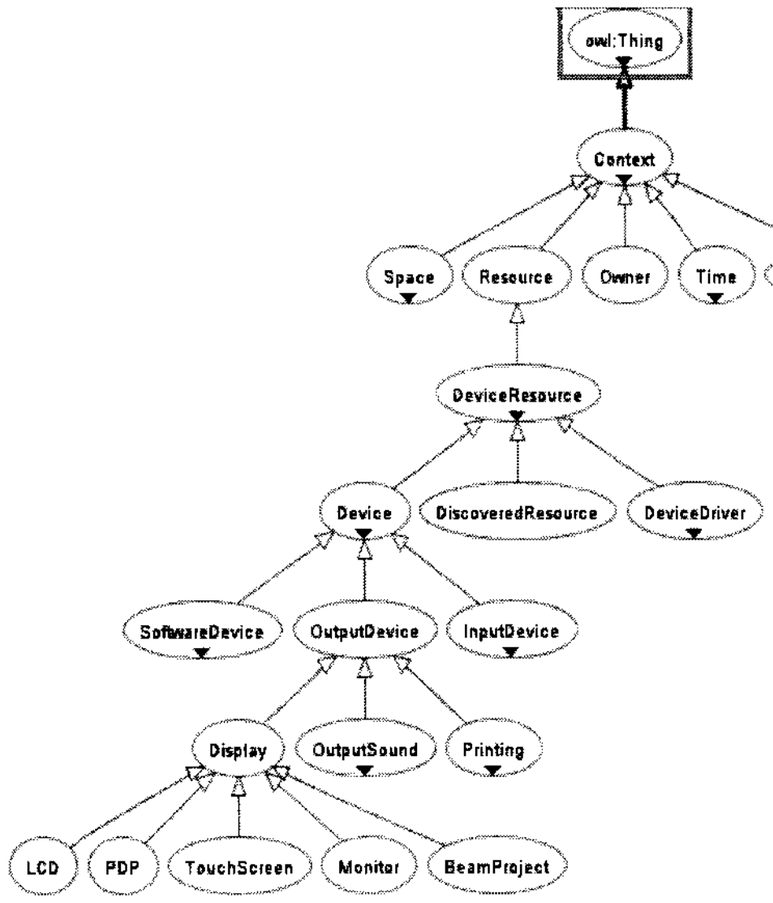


그림 10. 리소스 온톨로지 클래스 다이어그램
Fig. 10. Resource Ontology Class Diagram.

상황과 룰 그리고 본 연구에서 정의한 온톨로지를 이용하여 JESS를 통한 추론을 할 때에 더욱 더 사용자의 요구사항을 만족하는 추론에 접근 할 수 있을 것으로 기대된다.

마지막 세 번째는 사용자 정보를 온톨로지로 구성하는 것이다. 사용자 정보 온톨로지를 구성하는 이유는 본 연구가 UMO 환경에서의 상황 인지이며, UMO 환경의 특성상 개개인의 특성을 고려한 특화된 개별 서비스를 제공하기 위함이다. 본 연구에서는 사용자 정보를 온톨로지로 구성하기 위해서 사용자 정보를 목적과 역할에 맞게 다음과 같이 세분화한다. 첫번째는, 사용자의 선호도를 표현할 수 있는 User Preference 정보이다. User Preference는 사용자의 선호도 정보로 사용자가 어떤 것을 원하고 어떤 것을 원하지 않는지에 대한 정보를 가지고 현 상황에서 가장 원할 것으로 추측되는

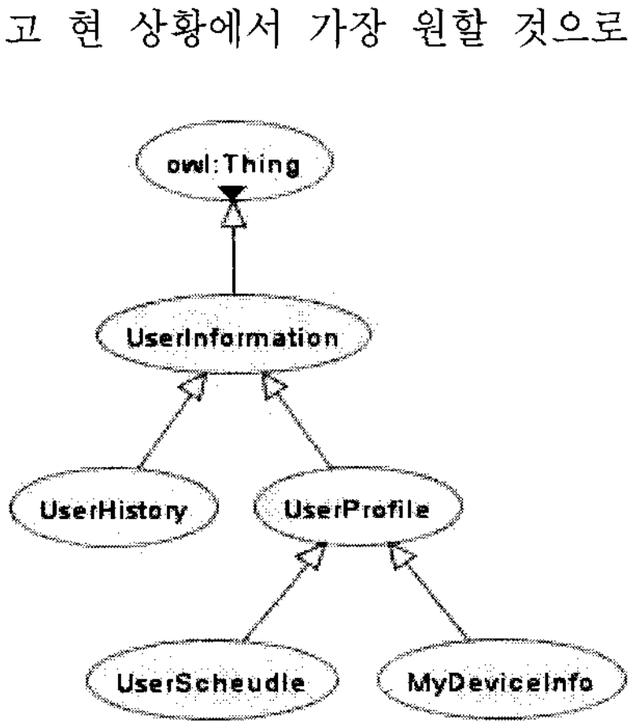


그림 11. 사용자 정보 클래스 다이어그램
Fig. 11. User Information Class Diagram.

서비스를 추론하기 위해서 사용된다. 두 번째와 세 번째는 각각 사용자 자신의 신상정보 또는 이력을 표현할 수 있는 User Profile과 사용자가 어떠한 일을 했는지 등에 관한 정보를 표현할 수 있는 User History이다. 이 두 정보들은 사용자의 신상정보와 이전 이력을 관리하므로 이를 이용한다면 모바일 환경에서 개인화된 최적의 서비스를 추론하기 위해서 매우 효율적으로 사용될 수 있다. 그림 11은 User Information을 위한 클래스 다이어그램이다.

2. 클래스의 구성

각 클래스는 Context라는 하나의 SuperClass의 SubClass의 형태로 존재하고 서로는 Parent - Child 관계를 가지게 되며, Context 클래스에 가까울수록 더 높은 수준의 클래스 즉, Upper 온톨로지가 된다. 모든 클래스는 Context 클래스로 수렴하여 마지막에는 이 Context 클래스에 모인 정보들이 하나의 컨텍스트, 즉 상황을 이룰 수 있게 된다. 룰과 함께 CIB에서 얻은 정보를 가지고 JESS에서 동작하여 상황인지를 수행하게 된다. 본 연구에서의 온톨로지의 구성은 아래의 표와 같다.

가. Context 클래스

Context 클래스는 UserInformation 클래스와 함께 최상위 Upper level을 구성하는 클래스이며, 지능공간 안에서 일어날 수 있는 대부분의 정보들과 의미적 대응관계를 가진 많은 서브클래스들을 포함하고 있다. Context 클래스는 UserInformation 클래스와 상호작용을 하여 정보를 공유한다.

- Space 클래스
- Time 클래스
- Communication 클래스
- Resource 클래스
- Activity 클래스
- TargetDeviceSchedule
- Owner

나. UserInformation 클래스

사용자의 정보를 관리하기 위해 표현된 클래스로 Context 클래스와 함께 최상위 level을 이루고 있다. 그리고 사용자의 선호도, 상태정보, 과거이력을 활용하여 보다나은 서비스를 제공하기 위한 클래스이다. UserInformation 클래스를 구성하는 클래스들은 Context 클래스의 정보를 이용가능하고, 반대로 Context 클래스 역시 UserInformation 클래스의 정보를 이용하는 것이 가능하다. 즉, 사용자 정보와 상황 정보를 이용하여 더 나은 단계의 추론을 제공하기 위함을 목적으로 하는 클래스이다.

- UserProfile 클래스
- UserHistory 클래스

다. Service 클래스

사용자가 이용 가능한 서비스를 미리 구성해 놓음으로써 서비스에 맞는 디바이스를 제공함에 있어서 조금 더 빠른 시간 안에 효율적인 방법을 모색하도록 하는데 도움이 될 수 있도록 구성해 놓은 클래스이다. 이 클래스 안에는 ServiceProfile을 두었고, 그 아래 Input, In_Output, Output 클래스를 두어서 Service를 Input을 주로 하는 것인지, Output의 역할이 큰지, 혹은 Input과 Output이 동시에 큰 역할을 하는지를 구별하도록 하였다. 또한 이 Service 클래스의 하위에 정의된 클래스들은 Context 클래스와 유기적으로 관계를 가지고 작용하도록 되어있으며, 서로를 참조하여 조금 더 효과적으로 사용자에게 서비스를 제공하고, 디바이스를 찾을 수 있도록 설계의 중심을 두었다.

- Input 클래스
- In_Output 클래스
- Output 클래스

V. 실험 시나리오

본 장에서는 서론에서 언급한 시나리오를 이용하여 제안한 Situation-Aware Service Engine이 상황을 고

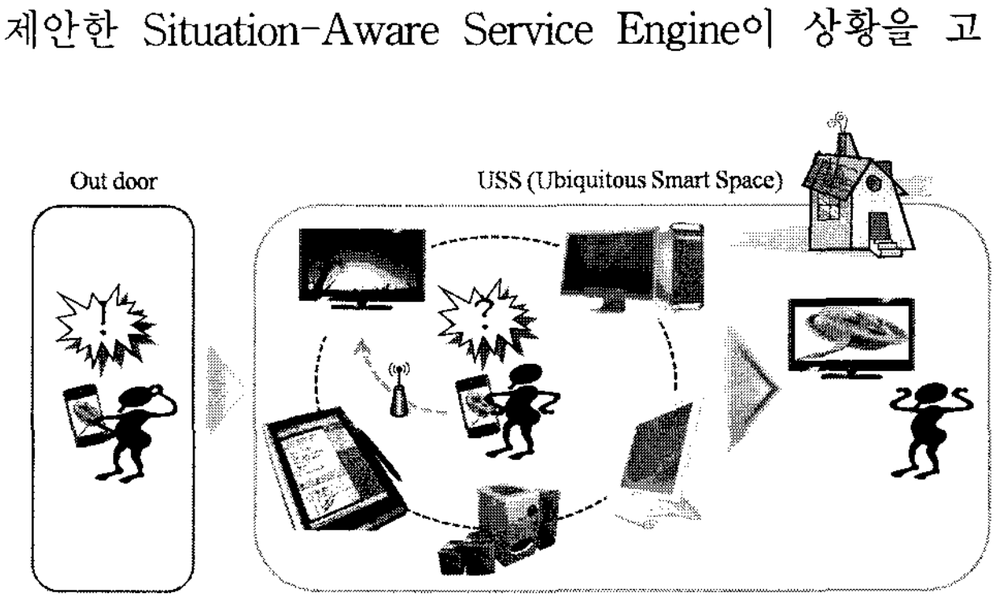


그림 12. 실험 시나리오
Fig. 12. Test Scenario.

려한 가용한 디바이스를 추천 해주는 예를 보인다.

주변의 여러 장치를 탐색한 결과 중 모니터에 관련된 정보는 다음과 같다고 가정하자.

```
<umo:Monitor rdf:about="&umo;monitor_01"
        umo:ID="monitor_01"
        umo:Monitor_Respond_Time="200"
        umo:Monitor_brightness="900"
        umo:Monitor_horizon_resolution="1024"
        umo:Monitor_is_CRT="false"
        umo:Monitor_is_LCD="true"
        umo:Monitor_size="20"
        umo:Monitor_vertical_resolution="1024"
        umo:annotation="monitor_04"
        umo:model="monitor_01"
        rdfs:label="monitor_01">
<umo:IsLocalResourceOf
        rdf:resource="&umo;LocalResource_01"/>
</umo:Monitor>
<umo:Monitor rdf:about="&umo;monitor_02"
        umo:ID="monitor_02"
        umo:Monitor_Respond_Time="100"
        umo:Monitor_brightness="300"
        umo:Monitor_horizon_resolution="2048"
        umo:Monitor_is_CRT="true"
        umo:Monitor_is_LCD="false"
        umo:Monitor_size="18"
        umo:Monitor_vertical_resolution="2048"
        umo:annotation="monitor_02"
        umo:model="monitor_02"
        rdfs:label="monitor_02">
<umo:IsLocalResourceOf
        rdf:resource="&umo;LocalResource_01"/>
</umo:Monitor>
```

또한, 사용자의 선호 정보는 개인의 사용이력 정보를 통해 다음과 같이 표현된다.

```
(User_Preference
(PreferredMonitor_Brand LG)
(PreferredMonitor_Size32)
(PreferredMonitor_Brightness 780))
```

마지막으로 시간을 고려한 모니터 디바이스를 추천하기 위해서 본 논문에서는 디스플레이 장치의 최대 해상도를 이용하여 시간을 고려한 모니터 디바이스를 추천하게 된다. 오전에 A라는 사람이 선호하는 모니터 정보를 찾는 룰과 그 결과를 나타내는 예는 다음과 같다.

Rule	(defrule MyRecommandationMonitor1 (Time (Date_type ?date_type)) (test (eq ?date_type AM)) (Monitor(ID?id) (Monitor_brightness ?brightness)) (test (< ?brightness 900)) => (?*a* add AM_Monitor) (printout t ?id crlf) (?*a* add ?id) (store ID ?*a*))
Result	AM_Monitor : monitor_02

VI. 결 론

본 논문에서 제안하는 상황인지 기반 디바이스 협업 시스템을 이용하여 리소스 제한적인 모바일 단말기의 문제점을 어느 정도 해결 할 수 있으리라 본다. 따라서 모바일 단말기는 사용자와 서비스간의 매개체 역할을 수행함에 있어서 좀 더 기능적으로 확장 가능하다고 볼 수 있다.

현재 우리는 디바이스 탐색 기술 및 사용자의 위치정보 탐색 기술을 본 시스템에 적용하기 위한 연구를 진행 중이다. 또한, 디바이스 협업 시스템을 좀 더 지능적으로 확장하기 위해 필요한 사용자의 선호도 정보 및 좀 더 다양한 상황 정보를 포함하는 프로파일 정보의 표준화된 표현 기법 및 효율적인 관리 기법에 대한 연구가 진행 중이다.

참 고 문 헌

[1] D. Bansal, J.Q. Bao, and W.C. Lee, "QoS-enabled residential gateway architecture", IEEE communications Magazine, vol. 41, April 2003.

[2] 이동환, "홈 네트워크 산업 현황 조사 연구 결과 보고", 2005 홈 네트워크 산업 현황과 전망 세미나, 2005.

[3] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Comm., vol. 8, no. 4, August 2001, pp. 10-17

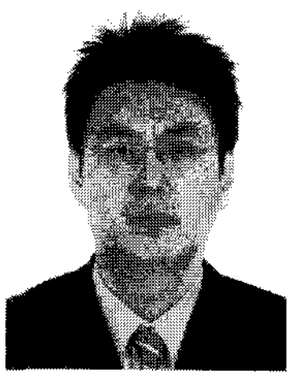
[4] D. Tennenhouse, "Proactive Computing", Comm. ACM, vol,. 43 no. 5, May 2000, pp. 43-50.

[5] D. Saha, A. Mykherjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer Vol.36, Issue 3, March 2003 pp.25-31.

[6] MC Lee, HK Jang, YS Paik, SE Jin and S Lee, "Device Collaboration Framework in Ubiquitous

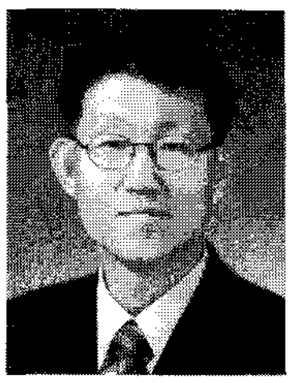
Environment: Celadon", SEUS-WCCIA'06, Gyeongju, Korea, April 2006.
[7] MC Lee, HK jang, SY Kim, YS Paik, SE Jin, S Lee, C. Narayanaswami, M.I.T. Raghunath and M.C.Rosu, "Celadon : Infrastructure for Device Symbiosis", The Seventh International Conference on Ubiquitous Computing 2005, Tokyo, Japan, September 2005.
[8] Harry Chen, Tim Finin, and Anupam Joshi, "An Ontology for Context-Aware Pervasive Computing Environments", In Proceedings of Workshop on Ontologies and Distributed Systems, in conjunction with IJCAI 2003 Conference, Acapulco, Mexico, August 2003.
[9] "About Context Broker Architecture", http://cobra.umbc.edu/about.html
[10] Anand Ranganathan and Roy H. Campbell, "A Middleware for Context-Aware Agent in Ubiquitous Computing Environments", Middleware 2003, Rio de Janeiro, Brazil, June 2003.
[11] JESS, http://herzberg.ca.sandia.gov/jess
[12] OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features
[13] OWL Web Ontology Language Semantics and Abstract Syntax, http://www.w3.org/TR/owl-semantics

저 자 소 개

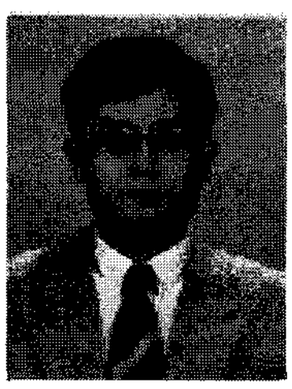


박 원 익(학생회원)
2004년 충남대학교 컴퓨터과학과 학사 졸업.
2006년 충남대학교 컴퓨터공학과 석사 졸업.
2007년~현재 충남대학교 컴퓨터공학과 박사 과정.
<주관심분야 : 지능형 추천 알고리즘, e-비즈니스, 유비쿼터스 컴퓨팅>

박 종 현(정회원)
1999년 우송대학교 컴퓨터과학과 학사 졸업.
2002년 충남대학교 컴퓨터과학과 석사 졸업.
2007년 충남대학교 컴퓨터과학과 박사 졸업.
2007년~현재 충남대학교 소프트웨어연구소 전임연구원.
<주관심분야 : XML, XQuery, Ontology, 분산 데이터베이스, 유비쿼터스 컴퓨팅, 웹정보시스템>



김 영 국(정회원)-교신저자
1985년 서울대학교 계산통계학과 학사 졸업.
1987년 서울대학교 계산통계학과 석사 졸업.
1995년 미국 버지니아대 컴퓨터과학과 박사 졸업.
1995년~1996년 핀란드 VIT, 노르웨이 SINTEF DELAB 방문연구원.
1996년~현재 충남대학교 컴퓨터공학과 교수.
<주관심분야 : 실시간데이터베이스, 모바일정보시스템, 전자상거래시스템>



강 지 훈(정회원)
1979년 서울대학교 계산통계학과 학사 졸업.
1981년 한국과학기술원 전산학과 석사 졸업.
1996년 한국과학기술원 전산학과 박사 졸업.
1996년~1998년 미국 버지니아대학교 컴퓨터과학과 방문교수.
2000년~2002년 충남대학교 정보통신원 원장.
1985년~현재 충남대학교 컴퓨터공학과 교수.
<주관심분야 : Semantic Web, 추론 알고리즘, XML, XQuery, 디지털 도서관, 데이터베이스 시스템, 웹 정보 시스템>