主題

# URC에서의 소프트웨어 로봇 기술

한국전자통신연구원 지능형로봇연구단 소프트웨어로봇연구팀
김 현, 이강우, 이주행, 강태근, 문애경, 서영호, 조준면

차 례



## 요 약

URC(Ubiquitous Robotic Companion)는 기존 로봇에 네트워크 및 정보 기술을 접목한 지능형 서비스 로봇의 새로운 개념으로서, 언제, 어디서나 나와 함께하며, 나에게 필요한 서비스를 제공하는 네트워크 기반 로봇이다. URC 개념이 구현되기 위해서는 유비쿼터스 네트워크 또는 센서 네트워크, 고성능 로봇용 서버 등과 같은 하드웨어 인프라가 구축되어 있어야 하며, 이러한 인프라 상에서 구동되는 소프트웨어 인프라가 필요하다. 소프트웨어 로봇은 이러한 소프트웨어 인프라에 해당하며, "유비쿼터스 네트워크 환경에서 언제 어디서나 상황에 맞는 정보와 서비스를 능동적으로 제공하는 새로운 개념의 지능형 소프트웨어"이다. 본 고에서는 이러한 소프트웨어 로봇의 개념과 이 개념을 지원하기 위한 시스템에 대해 논의한다.

Keywords : Ubiquitous Robotic Companion, network-based robot, software robot, context-awareness

## Ⅰ. 서 론

최근 산업용 로봇 시장이 포화되면서, 새로운 시장 창출을 위해 서비스 로봇이 활발히 연구 개발되고 있다. 서비스 로봇이 가전기기나 정보단말기와 같이 우리 생활 속에 들어오기 위해서는, 사용자가 저가로 쉽게 로봇을 구입하여 다양한 서비스를 받을 수 있도록 되어야 한다. 하지만

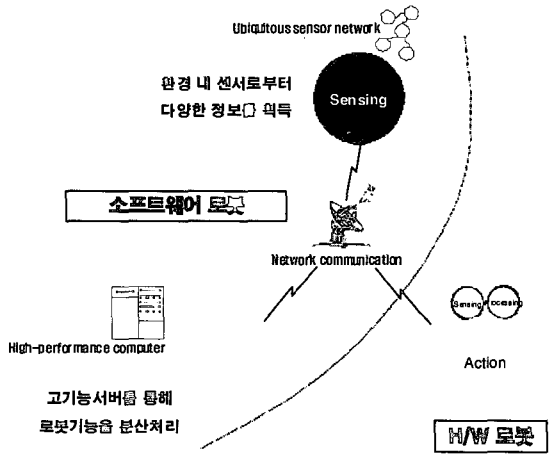


Fig. 1. URC의 개념

현재의 서비스 로봇은 가격에 비해 기능과 서비스가 단순하며 로봇 자체의 분명한 Killer Application을 갖고 있지 못하다. 이러한 배경 하에서 정보통신부에서는 로봇의 가격은 낮추는 반면 그 기능과 서비스는 다양화될 수 있도록 하기 위해 URC(Ubiquitous Robotic Companion)라고 하는 새로운 개념의 네트워크 기반 로봇을 개발 중에 있다.

일반적으로 로봇은 외부 환경을 센싱하고, 이를 바탕으로 판단하고, 이 판단에 따라 행동하는 세가지의 기능적 요소를 갖는다. URC에서 궁극적으로 추구하고자 하는 바는 로봇 자체에서 처리되던 이 세가지 필수 기능을 네트워크를 이용하여 분산하자는 것이고, 나아가서는 외부의 센싱 기능과 외부의 프로세싱 기능을 네트워크를 통해 충분히 활용할 수 있도록 하는 것이다 (Fig. 1). 즉, 로봇 자체의 센싱 기능을 늘려가기보다는 외부 환경에 내재된 센서 기능을 활용할 수 있게 하고, 기존 로봇의 프로세싱을 높이기보다는 원격지의 고기능 서버를 활용할 수 있게 하자는 것이다. 센싱 기능이 보완된다면, 외부 환경 및 사용자의 상황 인지가 훨씬 더 좋아질 수 있으며, 이에 따라 로봇이 보다 능동적으로 행동을 할 수 있게 된다. 또한 프로세싱의 제약이 극복됨으로써, 로봇의 기능이 그만큼 확장되고, 다양한 서비스가 가능해지며, 궁극적으로는 로봇의 지능이 높아질 수 있게 된다. 나아가, 사용자는 로봇을 떠나 있어도 원격지 서버에 접속하여 언제 어디서나 다양한 로봇 서비스를 받을 수 있게 된다. 결국 이러한 개념을 통해 최소한의 센싱 기능과 프로세싱 기능을 갖는 저가의 H/W 로봇만으로도 고도의 다양한 서비스를 받을 수 있으며, 나아가 단순 로봇 시장에 새로운 비즈니스 모델을 제시하여 이에 따른 부가가치 창출이 가능하게 된다.

## II. 소프트웨어 로봇의 개념 및 목표 기능

### 1. 소프트웨어 로봇의 정의

URC 개념이 구현되기 위해서는 유비쿼터스

네트워크 또는 센서 네트워크 등의 네트워크 인프라, 고성능 로봇용 서버 등과 같은 컴퓨터 하드웨어 인프라가 구축되어야 하며, 이러한 인프라 상에서 구동되는 소프트웨어 인프라가 필요하다. 즉, URC 개념에서 소프트웨어 인프라는 로봇과 고성능 서버와의 네트워크 통신을 통해 언제 어디서나 다양한 서비스를 제공할 수 있도록 지원한다. 우리는 이러한 소프트웨어 인프라의 개념을 소프트웨어 로봇이라고 명명하고 "유비쿼터스 네트워크 환경에서 URC 서버 상에 탑재되어 언제 어디서나 상황에 맞는 정보와 서비스를 능동적으로 제공하는 새로운 개념의 지능형 소프트웨어"로 정의한다. 소프트웨어 로봇은 기존 로봇의 공간적, 기능적 제한을 극복하고, 사용자 측면에서는 보다 저렴한 가격으로 다양한 서비스를 제공할 수 있도록 한다.

## 2. 소프트웨어 로봇의 목표 기능

소프트웨어 로봇은 고성능 서버와의 통신을 통해 기존 이동 서비스 로봇 (Mobile service robot)이 갖는 다음과 같은 몇가지 제약을 극복하는데 그 목표를 둔다.

**첫째, 나를 알아보고 내 말을 알아듣는다.**

이는 고도의 컴퓨팅 파워를 요구하는 음성인식 및 영상인식 기술 등과 연관되며, 기존 이동 서비스 로봇이 갖고 있는 가장 중요한 기능 중의 하나이다. URC에서의 로봇은 최소한의 프로세싱 능력을 갖는 저가의 H/W 로봇이라고 가정하고 있으며, 음성 및 영상인식 기능은 고성능 서버를 통해 지원받게 된다. 따라서 소프트웨어 로봇의 목표 기능 중의 하나는 로봇이 고성능 서버를 통해 보다 효율적으로 음성 및 영상인식을 할 수 있도록 하는 것이다.

**둘째, 언제 어디서나 사용자에게 서비스를 제공한다.**

URC에서 로봇 서비스는 URC 서버에서 제공되기 때문에 사용자는 H/W 로봇을 벗어나 있어도 자신의 소유한 정보 단말을 통해 서비스를 받을 수 있다. 즉, 소프트웨어 로봇은 기본적으로 네트워크를 통해 원격지의 서비스를 호출할 수 있어야 한다. 따라서 원격지의 서비스를 수행을 위해 필요한 프로그램 코드가 사용자의 정보단말(device)에 이동해 올 수 있어야 한다. 나아가 사용자의 정보단말에 제약이 없이 서비스를 제공받을 수 있어야 하기 때문에, 소프트웨어 로봇은 사용 정보 단말의 특성을 어떤 방법으로든 인식할 수 있어야 하며, 인식된 디바이스의 특성에 맞도록 서비스를 재구성할 수 있어야 한다.

**셋째, 사용자가 로봇에게 정보나 서비스를 요청할 때, 현재의 상황을 이해해서 그 상황에 맞는 서비스를 제공한다.**

URC에서의 로봇은 주변 환경에 내재된 센서 또는 응용 시스템으로부터 정보를 쉽게 얻을 수 있으며, 따라서 "서비스가 실행되는 환경"과 "서비스를 받는 사용자"에 대한 상황 인식이 훨씬 더 좋아질 수 있다. 일반적으로 사람이 작업을 수행할 때는, 현재의 상황을 잘 이해해서 적절히 반응하며, 필요한 시점에서 필요한 작업을 능동적으로 수행하지만 현재의 로봇은 그렇지 못하다. 상황을 인식한다는 것은 사용자가 관련된 많은 정보들을 일일이 명시적으로 입력해주지 않더라도 로봇이 상황을 이해하고 그 상황에 맞는 서비스를 제공할 수 있어야 한다는 것을 의미한다. 이를 위해서는 외부 환경과 사용자에 대한 상황 정보가 잘 모델링되고, 수집되고, 분배될 수 있어야 한다.

**넷째, 사용자의 요청이 없더라도 필요한 시점에서 필요한 정보와 서비스를 지능적이고 능동적**

으로 제공한다.

URC에서의 로봇은 사용자의 간접 명령이나 상황 인식을 통해 어떤 서비스가 제공되어야 하는지를 추론하여 사용자에게 필요한 서비스를 필요한 시점에 스스로 제공할 수 있어야 한다. 따라서 소프트웨어 로봇은 정보를 지식으로 표현하여 저장하고 저장된 지식을 활용하기 위한 수단을 제공할 수 있어야 한다. 나아가 항상 같은 상황에서 같은 서비스가 되지 않기 위해서 이 지식은 성장할 수 있어야 하며, 서비스 역시 새로운 서비스로 진화될 수 있어야 한다.

본 고에서는 이러한 소프트웨어 로봇의 목표 기능 중 특히, 상황인식과 관련한 시스템 미들웨어인 CAMUS (Context-Aware Middleware for URC Systems)에 대해 논의하고자 한다.

## III. CAMUS의 개념 모델

소프트웨어 로봇 관점에서의 실세계는 물리공간과 가상공간으로 구성된다. 물리공간은 실제적인 물리자원들로 구성되어 있으며, 사람은 이러한 물리객체와의 상호작용을 통해 일을 한다. 가상공간은 물리공간을 추상화 것으로써, 물리객체들을 매핑한 전산 자원들로 모델링된다. 소프트웨어 로봇은 사람을 대신하여 가상공간과의 상호작용을 통해 일을 한다. URC 개념에서의 로봇은 물리공간에 존재하는 물리객체 중의 하나가 되며, 가상공간 상의 한 객체로 매핑되어 소프트웨어 로봇에 의해 제어된다.

이러한 배경에서 CAMUS는 상위 개념에서 Fig. 2와 같은 추상화된 개체들로 모델링된다.

World는 물리공간을 추상화한 가상공간으로서 여러 개의 Environment를 갖는다. Environment는 임의의 한정된 공간을 추상화한 것으로 컴퓨터와 통신이 가능하고 고유의 식별자를 갖는 각종 센서와 장치들로 구성된다. Service는 Environment 내에 존재하는 다양한 자원의 물리적 기능을 가상공간에서 수행할 수 있는 컴퓨터 프로그램 모듈이다.

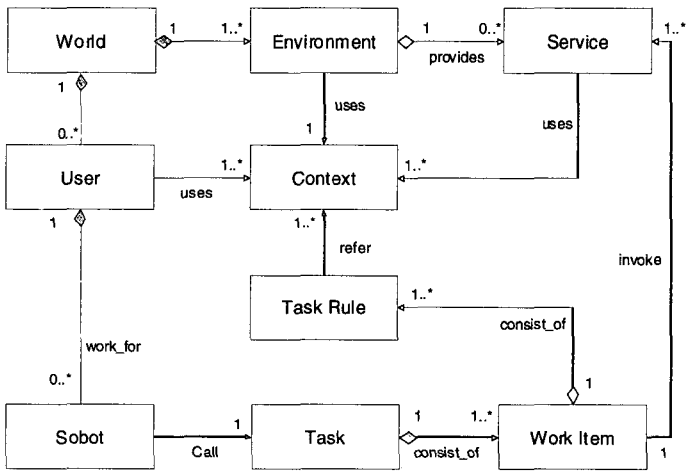


Fig. 2. CAMUS 개념 모델

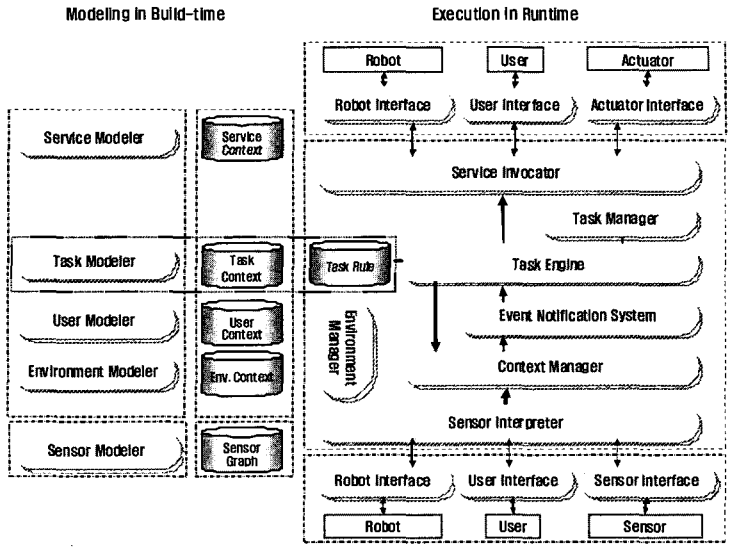


Fig. 3. CAMUS 시스템 구조도

사용자 역시 식별자, 프로파일 정보, 선호사항 정보 등을 갖는 가상공간 상의 사용자로 매핑된다. 실세계의 사용자는 Sobot이라는 인터페이스를 통해 가상공간에 접근한다.

Sobot은 실세계의 로봇과 같은 역할을 하며, World의 상태를 받아들이고 이에 따른 작업을 수행하여 World의 상태를 변화시킨다. 이러한 과정은 Task의 수행에 의해 이루어진다.

Task는 사용자의 요청이나 필요한 상황에 따라 수행해야 할 일련의 행위의 집합으로서, CAMUS에서의 작업 수행 단위이다. Task는 하위에 많은 Work item들을 가지며, 각 Work item들은 Task rule에 의해 기술된다. Task는 주어진 상황에 따라 적합한 Work item을 선택하고, 그에 해당하는 Task rule에 따라 Service를 호출하는 방식으로 동작된다. Task Rule은 Task 내의 Work Item 수행을 기술한 규칙이다. 사용자의 명령, 환경 변화 등의 상황에 따라 행위를 발생시키기 위해 필요한 규칙이다.

Context는 Task가 특정한 상황에 맞게 수행함에 있어서 필요한 제반 정보 및 지식을 의미한다. 다양하고 복잡한 Context의 의미와 그 상관관계를 이해하여 Task를 수행하는 것은 소프트웨어 로봇의 지능적이고 능동적인 특성이 된다. Context의 추출은 기본적으로 특정 Environment의 데이터를 기술한 사실 및 이들 간의 관계를 조합하여 새로운 정보를 얻을 수 있는 추론 규칙에 기반한다.

## IV. CAMUS의 구조

Fig. 3은 본 논문에서 제시하고자 하는 CAMUS의 구조도를 보여준다. CAMUS는 물리공간에 대한 가상공간 모델링 부분과 작업 실행 부분으로 나뉘어진다.

### 1. 가상공간 모델링

가상공간 모델링은 크게 센서 모델링(Sensor Modeling), 서비스 모델링(Service Modeling), 환경 모델링(Environment Modeling), 사용자 모델링(User Modeling) 및 작업 모델링(Task Modeling) 등으로 구성된다.

센서 모델링은 물리 공간의 센서를 가상 공간으로 매핑하고, 이들 센서 정보로부터의 Context를 추출하여 수행 엔진(Task Engine)에 제공함으로써, 능동적인 서비스를 제공할 수 있도록 지원한다. 센서 모델링의 결과는 서비스로 작성되며, 작업에 의해 서비스로 호출된다.

서비스 모델링은 장치 제어, 음성 처리, 일정 관리 등과 같이 CAMUS의 작업 단위로부터 수행될 서비스(컴퓨터 프로그램 모듈)들에 대한 인터페이스와 이들 구현 코드에 대한 관리를 지원한다. 본 논문에서 CAMUS에 의해 호출되는 서비스는 CORBA 객체로 가정하였다. 따라서 서비스 인터페이스는 IDL(interface definition language)로 정의되며, 구현 코드는 CORBA Stub/Skeleton code를 포함하는 프로그램 코드들의 집합이다.

환경 모델링은 물리공간의 한정된 영역과 해당 영역에 존재하는 가용 자원에 대한 모델링이다. 네트워크로 통신할 수 없는 센서나 기기 등은 가상공간 상에서 의미가 없다. 따라서 이러한 센서나 기기는 궁극적으로 서비스의 형태로 시스템과 상호작용을 한다. 따라서 환경 모델링이 끝나고 나면, 서비스 모델링 시점에 등록된 코드들이 실행되어 서비스를 준비하게 된다.

사용자 모델링은 CAMUS에 등록된 사용자 정보에 대한 모델링이다.사용자 정보는 인적 프로파일 정보 및 선호사항 정보 등을 포함한다.

작업 수행을 위해는 작업 모델링이 이루어져야 한다. 이는 기존에 모델링된 환경 정보를 이용하여, 특정 작업을 수행하기 위해 필요한 특정 상황정보를 모델링하고, 해당 작업 수행에 필요한 작업 규칙 (Task Rule)을 작성하는 것이다.

CAMUS에서 사용하는 모든 상황 정보는 Universal Data Model (UDM)을 통해 표현된다. UDM에서는 모든 상황 정보를 노드와 노드 사이의 연관으로 표현한다. 노드는 가상공간에서 개체를 표현하며 사람, 장소, 작업, 서비스 등이 된다. 모든 노드는 각각 고유한 식별자와 타입을 갖는다. 또한 노드의 특정 타입으로 "valued" 노드가 있으며, 이는 임의의 Java 객체로부터 값을 얻어올 수 있는 노드이다. 연관은 노드 사이의 관계를 기술하며, 방향성을 갖는 화살표로 표현되며 그 의미를 표현하는 이름을 갖는다. 화살표가 시작되는 노드를 시작 노드(from node)로, 화살표가 끝나는 노드를 종료 노드(to node)라 부른다. Fig. 4는 UDM을 통한 모델링의 예를 보여준다.

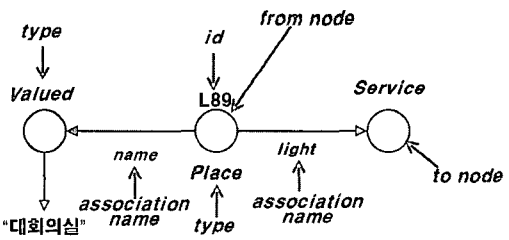


Fig. 4. Universal Data Model

CAMUS에서의 작업 규칙은 ECA(Event-Condition-Action) 규칙 표현에 따른다. 이러한 작업 규칙은 PLUE(Programming Language for Ubiquitous Environment)라고 하는 프로그래밍 언어를 통해 기술된다. 이 언어는 사용자가 보다 쉽게 작업 규칙을 기술하기 위해 만들어진 언어이다. 다음은 PLUE로 기술된 간단한 작업 규칙의 예를 보여준다.

```
on ( $place.temperature::ValueChanged e )
if (e.value>$place.resident.preferred_temp.high ) {
```

```
        $place.air_conditioner.turnOn();
    }
```

"on" part에는 작업 규칙의 수행을 유발하는 이벤트 연산식(event description)이 기술되고, "if"는 해당 작업 규칙이 수행되기 위한 조건을 기술하고, if 내부는 "action"에 해당하는 부분으로써, 명시된 조건이 만족하는 경우 수행될 서비스를 기술한다. 이 예에서는 온도값이 변화되고, 사용자가 원하는 온도보다 높으면, 에어컨을 작동시키는 간단한 규칙을 보여준다. 변수에 대한 값들은 Context 정보로부터 얻어진다.

## 2. 작업 수행

그림 5는 CAMUS에서의 작업 수행에 관련된 정보의 흐름과 시스템 구성 모듈을 보여준다.

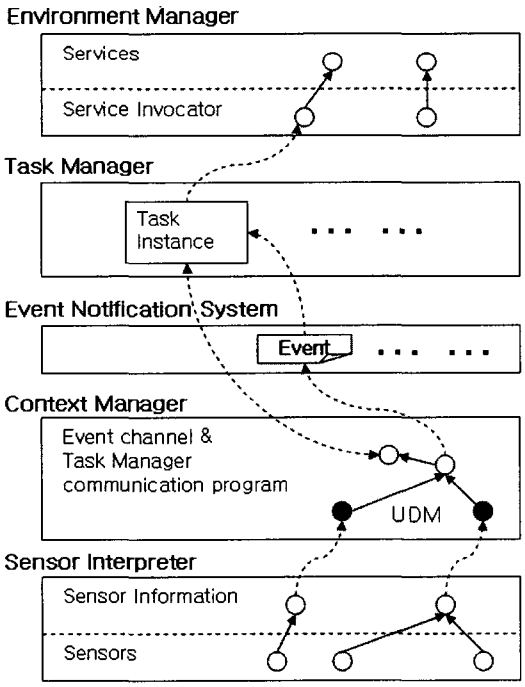


**Fig. 5. 작업 수행을 위한 정보 흐름**

센서 해석기(Sensor Interpreter)는 환경 내의 물리센서, 응용 시스템 또는 사용자 명령 등의 입력 정보를 상황정보에 맞도록 가공하고 이 정보를 상황 관리자에게 전달하는 모듈이다. 사용자로부터의 음성정보, 영상정보, 온도/습도 센서, 사용자 일정정보 등이 모두 센서 정보가 될 수 있다. 특히, 음성명령의 경우, 센서 해석기는 음성인식기 (Voice Recognizer)를 통해 사용자의 음성명령을 문자열로 변환하고 상황정보에 맞도록 가공하여 이를 상황 관리자 에게 넘겨준다.

상황 관리자(Context Manager)는 센서 해석기로부터 전달된 상황정보를 별도의 저장소에 관리한다. 저장소에는 상황정보가 UDM으로 표현되어 관리된다. 또한 상황관리자는 상황정보에 변경이 발생했을 때, 그 Event를 이벤트 통지 시스템(Event Notification System)으로 전달하고, 작업엔진(Task Engine)에서 작업을 수행하는데 필요한 상황정보를 제공하는 역할을 한다.

이벤트 통지 시스템은 센서나 기타 외부 서비스를 통해 상황 관리자에 의해 전달되는 이벤트를 이를 원하는 작업에게 전달하는 역할을 한다. 이벤트 통지 시스템은 CORBA의 Notification Service를 활용하여 개발된다.

작업 관리자(Task Manager)은 개별 작업을 기동시키고, 수행중인 작업 프로세스를 관리하는 역할을 한다. 작업 엔진(Task Engine)은 상황에 따른 실제 작업을 수행시킨다. 작업 엔진은 해당 작업에 대한 상황 정보(Fact)와 작업 규칙(Rule)을 추론할 수 있는 추론 엔진을 갖는다. 본 시스템에서는 JESS를 추론 엔진으로 사용한다.

마지막으로, 환경관리자 (Environment Manager)는 작업 엔진에 의해 호출되는 서비스의 실행을 담당한다.

## 3. 사용자 인터페이스

사용자와 CAMUS 간의 상호작용은 소봇

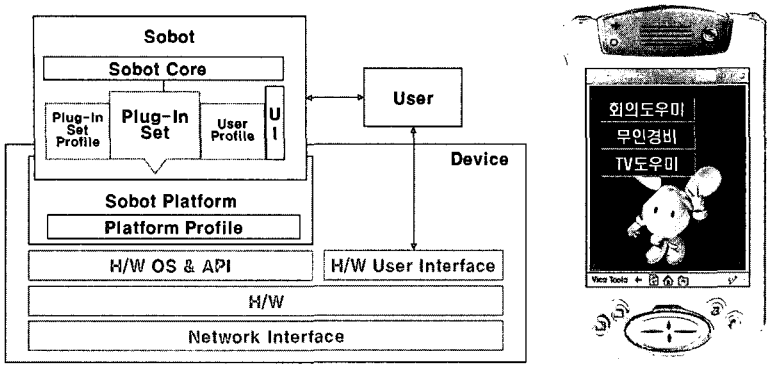


Fig. 6. 소봇의 구조 및 GUI

(Sobot)에 의해 이루어진다. 소봇은 사용자의 명령을 받아 시스템에 전달하거나 시스템으로부터의 메시지를 사용자에게 전달하는 역할을 한다. Fig. 6은 소봇의 구조 및 GUI를 보여준다. 소봇은 소봇 플랫폼(Sobot Platform)이 탑재된 다양한 정보단말기 상에서 구동하며, 소봇 코어(Sobot Core)와 플러그인(Sobot Plug-In)들로 구성된다. 소봇 코어는 소봇이 구동되기 위한 최소 핵심부이며, 소봇 플랫폼과의 통신, 입출력 처리, 플러그인 다운로드 등의 기능을 한다. 또한 플러그인의 프로파일 정보를 관리하고 서비스 중에 습득한 지식 및 사용자 선호사항을 관리한다. 플러그인은 특정 서비스와 연관되어 클라이언트에서 요구되는 각종 프로그램 모듈들로서, 소봇 코어에 의해 필요한 시점에 다운로드 된다.

## V. 결 론

URC는 기존의 서비스 로봇이 산업화, 대중화되기 위한 하나의 대안으로 제시되었다. 이는 최소한의 센싱 기능과 프로세싱 기능을 갖는 저가의 H/W 로봇만으로도 고도의 다양한 서비스를 받을 수 있도록 하고자 하는 목적을 갖는다. CAMUS는 로봇이 서비스하는 환경을 모델링하기 위한 방안을 제시하며, 환경 내 상황정보를 이용하여 로봇이 보다 능동적인 서비스를 하기 위한 시스템 하부 엔진을 제공한다. 현재는 개발된 시스템을 이용해 실제 가정 환경에서 로봇이 상황 인지 기반 서비스를 하기 위한 몇가지 시나리오를 구현 중에 있다.

**김 현**

1984년 : 한양대학교 기계설계학과 학사
1987년 : 한양대학교 기계설계학과 석사
1997년 : 한양대학교 기계설계학과 박사
1998년 ~ 1999년 : 한양대학교 산업공학과 겸임교수
1990년 ~ 현재 : 한국전자통신연구원 소프트웨어로봇 연구팀장, 책임연구원

<관심분야> Intelligent System, Distributed Computing, Context-Awareness, Engineering Knowledge Management