5

네트워크 로봇을 위한 로봇 소프트웨어 플랫폼에 대한 연구

한국전자통신연구원 ▌정승욱 · 이승익 · 김성훈

1. 서 론

지능형 서비스 로봇은 일반 가정 혹은 사무실 환경에서 네트워크 시스템과 연계하여 인간과 상호작용하면서 주어진 환경에 맞는 적절한 태스크를 수행하는 로봇이다[1]. 지능형 서비스 로봇은 정해진 일을 반복 수행하는 산업용 로봇과는 달리 변화하는 환경에 능동적으로 대처하며 인간과 밀착된 서비스를 제공하는 형태로 발전되어가고 있으며, 지난 100년간 자동차와 컴퓨터가 인간 생활에 큰 변화를 가져다 준 것처럼 향후 10~20년 내에 인간의 일상생활에 커다란 변화를 가져다 줄 것으로 예측되고 있다[2,3].

최근 눈부시게 발전하고 있는 IT(Information Technology) 기술을 바탕으로 가사, 오락, 공공 서비스와 같은 다양한 서비스를 제공하는 네트워크 로봇(ubiquitous-로봇, u-로봇)에 대한 관심이 고조되고 있다. 네트워크 로봇은 IT기반 지능형 서비스 로봇으로서, 기존의 독립형 로봇에서 벗어나 인간과 서로 상호작용하면서 시간, 장소, 상황에 제한되지 않고 유비쿼터스(Ubiquitous) 환경 내의 다양한 센서 정보와 연계하여 사용자에게 능동적으로 서비스를 제공한다. 유비쿼터스 환경 속에서 언제 어디서나 사용자가 원하는 서비스를 능동적으로 제공 가능한 네트워크 로봇은 기존 로봇 개념에 유비쿼터스 네트워크와 정보 기술을 결합한 지능형 서비스 로봇의 새로운 개념으로, 물리적인 로봇뿐만 아니라 지능형 소프트웨어 로봇 등을 포함한다. 지능형 소프트웨어 로봇은 유비쿼터스 네트워크 환경에서 실세계 객체와 통신하며 언제, 어디서나 상황에 맞는 정보와 서비스를 능동적으로 제공하는 새로운 지능형 소프트웨어로서 일반 로봇과 마찬가지로 환경 정보를 센싱하고 그 정보를 기반으로 할 일을 계획하며, 그 결과를 수행하거나 그 행위를 로봇이나 주변의 다른 기기에 대행하도록 한다[4,5].

정부에서는 2004년도부터 국민 소득 2만 달러 시대를 여는 핵심 기술 및 신 성장 동력의 하나로 로봇을 선정하고 정부, 학계, 연구소, 산업계가 공동으로 URC (Ubiquitous Robotic Companion)라는 개념으로 네트워크 기반의 지능형 서비스 로봇 사업을 추진하고 있다. URC는 "언제 어디서나, 나와 함께 하며, 나에게 필요한 서비스를 제공하는 로봇"으로 간단하게 정의할 수 있다. 일견 단순해 보이는 이 정의는 많은 내용을 함축하고 있지만 가장 중요한 것은 URC 로봇이 유선 혹은 무선 통신이 가능한 각종 외부 디지털 디바이스와 상호 연동할 수 있어야 한다는 것이다. 이를 통하여 환경 인식이나 음성 인식 등과 같이 로봇이 수행해야 할 기능을 외부 디바이스에 분담시킴으로써 로봇의 하드웨어 구성을 단순화시켜 제작 원가를 절감하고, 날씨나 교통 정보와 같이 로봇 단독으로는 획득할 수 없는 정보 등을 이용하여 보다 다양한 서비스를 할 수 있다는 점이다. 즉, URC 로봇을 기존 서비스 로봇보다 향상된 서비스를 지원하면서도 더 싸게 만들 수 있게 된다[6].

RUPI(Robot Unified Platform Initiative)는 네트워크 로봇 소프트웨어 플랫폼간의 상호호환성, 다양한 통신 및 정보기기와의 상호운용성, 이종 통신망과의 상호접속성을 갖는 로봇 소프트웨어 표준 플랫폼 및 제반 표준 규격을 말한다. RUPI를 적용하여 개발된 로봇은 다양한 종류의 로봇 및 정보기기와 상호 연동이 가능하게 되어 로봇 소프트웨어 개발 및 로봇산업 경쟁력 제고에 큰 도움이 될 것으로 기대된다[7].

본 고에서는 RUPI 규격 중 로봇 내부 태스크 및 콘텐츠 실행 미들웨어의 참조 구현 시스템인 uROSE(urc RObot Service development Environment)에 대해 기술한다. 본 고의 2장에서는 RUPI에 대해 소개하고, 3장에서는 로봇 소프트웨어 플랫폼의 기존 연구 동향에 대해 기술한다. 4장에서는 uROSE에 대해 기술하며 5장에서 결론을 맺는다.

2. RUPI

RUPI는 네트워크 로봇이 제공하는 복잡한 서비스를

쉽고 편리하게 개발하기 위한 각종 소프트웨어의 표준 규격을 제정하고 이를 참조 구현하는 것을 목표로 한다. RUPI는 로봇 소프트웨어 컴포넌트의 재사용성 및 상호호환성, 다양한 정보 기기와의 상호 운용성, 이종 통신망과의 상호 접속성을 갖는 지능형 로봇의 소프트웨어 규격 및 구현 모델이다[7].

2.1 RUPI 추진 경과

RUPI는 2005년 12월 경 정보통신부 하반기 전략회의에서 그 필요성이 제기되었고, 10여 차례에 걸친 TFT (Task Force Team) 및 RUPI 추진위원회 회의를 통해 RUPI 기본 계획안이 2006년 5월경에 확정되었다. 이후 URC 서버와 로봇 간의 통신 프로토콜, 표준 인터페이스 등 2종의 RUPI 1.0규격이 제정되었으며, 2006년도 URC 시범사업에 RUPI 1.0 규격의 참조 구현 시스템이 적용되어 그 실용성을 검증하였다.

2007년도에 RUPI 1.0에 대한 로봇 플랫폼, 서비스 기업의 의견을 반영하여 RUPI 1.2를 제정하고, URC 시범사업 및 국민로봇사업에 적용하였다. RUPI 1.2는 기업체의 의견 중 시급성과 개발시간 등을 고려하여 기존의 프로토콜 보완 1종 및 신규 1종, 권장 프로토콜 1종 등 총 3종의 규격으로 구성되었으며, 2007년 시범사업 및 2기 국민로봇사업단 참여 기업을 대상(10개 기업, 27명)으로 기술 교육을 실시하였다.

로봇 업체가 우선적으로 필요한 요구사항 위주로 작성된 RUPI 1.2와 병행하여, RUPI 실무지원반은 6개 워킹그룹(로봇 내부 S/W, URC 서버 S/W, 네트워크, 로봇 콘텐츠, 통합 개발 환경, 응용 컴포넌트)을 구성하여 RUPI 2.0 규격 28종을 작성하였다. 2007년 10월경 RUPI 2.0 규격 초안 공개 및 공청회가 열렸으며, 현재 RUPI 2.0 규격 3차 버전까지 공개되었다.

2.2 RUPI 규격

RUPI 규격은 로봇 소프트웨어가 따라야 하는 최소한의 요구 사항을 의미론적으로 기술한 것과 컴포넌트 및 인터페이스 등에 대한 기술 내용을 의미하며, RUPI 구현 모델은 RUPI 규격을 만족하는 소프트웨어 모듈로서 바로 적용할 수 있는 구현 물을 의미한다. 그림 1은 RUPI 규격의 기본 구조를 나타낸 것이다.

RUPI 규격은 크게 RUPI 서버, RUPI 클라이언트, 통신 및 응용컴포넌트 인터페이스, 개발도구 등으로 구성되어 있다. RUPI 클라이언트 규격은 로봇 내부 소프트웨어에 대한 표준 규격이며, RUPI 서버 규격은 RUPI 클라이언트를 관리하고 서비스를 제공하는 서버에 대한 표준 규격이다. RUPI 규격은 네트워크를 통한 다양한 서비스를 제공하기 위한 RUPI 서버 소프트웨어, 네트워크 프로토콜, 로봇 콘텐츠를 함께 다룬다는 측면에서 국내외의 다른 기술과 차별화된다고 볼 수 있다. RUPI에서 대상으로 하는 로봇은 필요시에만 RUPI 서버와 연결하여 도움을 받고 평상시에는 독립적인 기능을 수행하는 Rich-Client 로봇과 RUPI 서버의 도움을 받아 대부분의 기능을 수행하는 Thin-Client 로봇으로 구분된다. RUPI는 고기능의 리치 클라이언트(Rich-Client) 로봇에서부터 저사양의 씬 클라이언트(Thin-Client) 로봇까지 다양한 로봇을 대상으로 하고 있어, 다른 기술과 차별화뿐만 아니라 폭넓은 선택에 의한 로봇 시장의 다양화를 유도할 수 있다.

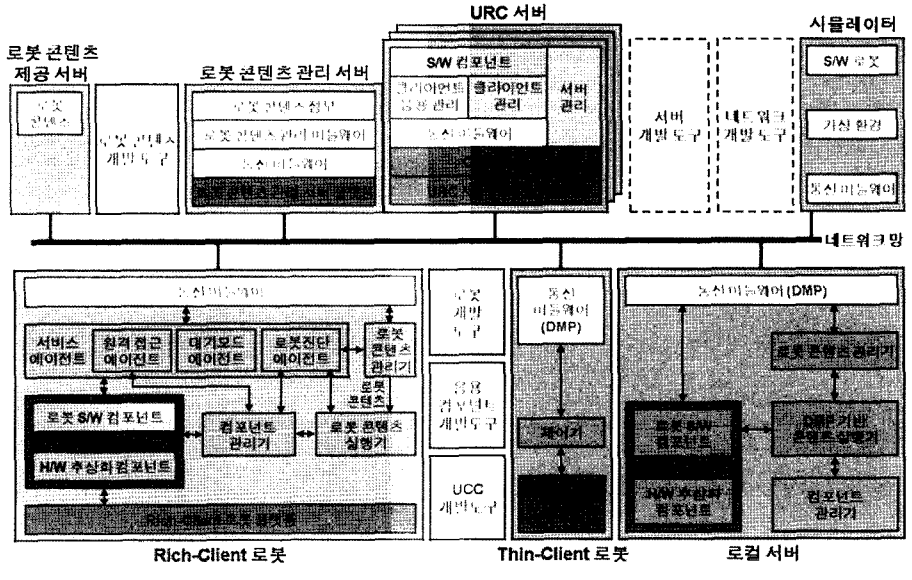

그림 1 RUPI 규격 기본 구조

다음은 RUPI 규격의 현황을 요약한 것이다.

o RUPI 1.0
- RUPI 서버와 로봇 간의 통신 규격 : 원격 객체 호출 방식을 기반으로 한 로봇 서버와 로봇 간의 통신 방식에 대한 규격을 정의함
- RUPI 서버와 로봇 간 통신을 위한 표준 인터페이스 : RUPI 서버에서 로봇의 기능을 호출하거나 로봇에서 서버의 기능을 호출할 때 필요한 표준 인터페이스를 기술함

o RUPI 1.2
- URC 서버/클라이언트 통신 프로토콜 규격 : 유비쿼터스 환경에서 상호연동을 지원하는 통신 프로토콜에 대한 규격을 정의함
- URC 로봇의 응용 S/W를 위한 표준 인터페이스 규격 : 네트워크를 통해 URC 로봇을 제어하거나 URC 서버와 클라이언트가 데이터를 주고받기 위한 인터페이스에 대한 규격을 정의함
- 디바이스 맵 기반의 URC 서버/클라이언트 통신 프로토콜 규격 : 네트워크를 통해 URC 서버에서 콘텐츠(동작, 음성, 영상)를 받고 이를 실행하기 위한 URC 서버와 로봇 간의 통신 규격을 정의함

o RUPI 2.0
- 로봇 내부 S/W 규격 8종
  - 리치 클라이언트 로봇의 로봇 소프트웨어 컴포넌트 : 리치 클라이언트 로봇을 위한 표준 컴포넌트 모델에 대한 규격을 정의함
  - 리치 클라이언트 로봇의 하드웨어 추상화를 위한 디바이스 컴포넌트 : 로봇에 사용되는 센서 및 액추에이터를 접근하기 위한 인터페이스 규격을 정의함
  - 리치 클라이언트 로봇의 로봇 구성 파일 : 로봇의 구성 정보를 기술하는 방식에 대한 규격을 정의함
  - 리치 클라이언트 로봇의 로봇 소프트웨어 컴포넌트 관리 : 리치 클라이언트 로봇을 위한 컴포넌트 관리 및 이를 위한 동작 구조에 대한 규격을 정의함
  - 리치 클라이언트 로봇의 대기 모드 에이전트 : 대기모드 지원을 위한 서비스 에이전트(Agent)에 대한 규격을 정의함
  - 씬 클라이언트 로봇용 로컬 서버의 로봇 응용 소프트웨어 : 씬 클라이언트 로봇용 제어 서버인 로컬 서버(Local Server) 에서 실행되는 로봇 응용 소프트웨어에 대한 규격을 정의함
  - 씬 클라이언트 로봇의 하드웨어 추상화를 위한 디바이스 컴포넌트 : 씬 클라이언트 로봇의 디바이스의 동작 방법 및 접근 인터페이스에 대한 규격을 정의함
  - 씬 클라이언트 로봇을 위한 서버 관리 : 씬 클라이언트 로봇을 위한 서버 관리 구조에 대한 규격을 정의함
- URC 서버 S/W 규격 4종
  - URC 서버 가용성 : URC 로봇 서버 플랫폼의 가용성을 위한 소프트웨어 구조에 대한 규격을 정의함
  - URC 로봇 클라이언트 관리 : URC 로봇의 원격 관리 및 원격 진단에 관한 규격을 정의함
  - URC 원격 로봇 응용 수행을 위한 서버 서비스 에이전트 : URC 원격 로봇 응용이 사용하는 서버 서비스 에이전트(SSA; Server Service Agent)의 프로파일, 기능, 구조에 대한 규격을 정의함
  - URC 로봇 클라이언트 응용 관리 : URC 로봇 클라이언트 응용이 사용하는 인터페이스 및 실행 환경에 대한 규격을 정의함
- 통신 프로토콜 규격 3종
  - URC 서버/클라이언트 통신 프로토콜(RPC 계층) : 리치 클라이언트 로봇과 URC 서버 간의 통신 프로토콜 규격 중 원격 함수 호출(Remote Procedure Call: RPC)에 대한 규격을 정의함
  - URC 서버/클라이언트 통신 프로토콜(Transport 계층) : URC 서버와 이기종 리치 클라이언트가 상대방을 찾고, 데이터를 신뢰성 있고 효율적으로 교환하는데 필요한 데이터 형식과 통신 프로토콜에 대한 규격을 정의함
  - 씬 클라이언트 로봇을 위한 디바이스 맵 기반의 URC 서버/클라이언트 통신 프로토콜 : 네트워크를 통해 URC 서버에서 서비스 콘텐츠(동작, 음성, 영상)를 받고 이를 실행하기 위한 URC 서버와 로봇 간의 통신 규약을 정의함
- 로봇 콘텐츠 규격 2종
  - 로봇 콘텐츠 관리 서버 : URC 로봇이 다양한 로봇 콘텐츠를 제공받기 위해서 로봇 콘텐츠를 관리하는 서버에 대한 규격을 정의함
  - 리치 클라이언트 로봇을 위한 로봇 콘텐츠 : 로봇 플랫폼에서 운영되는 로봇 콘텐츠와 로봇 콘텐츠의 실행 방식에 대한 규격을 정의함
- 통합 개발 환경 규격 5종
  - 씬 클라이언트 로봇의 메타모델과 콘텐츠 모델링 도구 : 씬 클라이언트 로봇의 모델 설계의

기반이 되는 메타 모델 및 모델링 도구에 대한 규격을 정의함
- 타임 라인 기반의 콘텐츠 모델링과 모션 편집 도구 : 로봇 콘텐츠를 제작함에 있어서 실행시의 상태에 영향을 받지 않는, 즉 정적 콘텐츠인 로봇 모션 콘텐츠 및 편집 도구에 대한 규격을 정의함
- 리치 클라이언트 로봇 소프트웨어의 개발환경 : 컴포넌트와 아키텍처에 기반하여 모델링을 수행하고 개발하는 방법으로, 소프트웨어를 개발할 때 필요로 하는 개발도구의 구성 요소와 개발절차에 관한 규격을 정의함
- 통합개발환경 내의 소프트웨어 컴포넌트 및 개발 도구의 상호 연동 : 네트워크 기반의 로봇 S/W 통합개발환경을 구성하는 번들에 대한 상호 연동 규격을 정의함
- 동적 콘텐츠 제작을 위한 씬 클라이언트 로봇의 액션 트리와 편집 방법 : 씬 클라인언트 로봇을 위한 정적인 콘텐츠를 타임라인 기반으로 개발하기 위한 규격을 정의함

– 응용 컴포넌트 규격 6종
- URC 로봇을 위한 음성 인식 컴포넌트 : 음성 인식에 사용되는 데이터 형과 컴포넌트를 구성하기 위해 필요한 인터페이스에 대한 규격을 정의함
- URC 로봇을 위한 음성 합성 컴포넌트 : 음성 합성에 사용되는 데이터 형과 컴포넌트를 구성하기 위해 필요한 인터페이스에 대한 규격을 정의함
- URC 로봇을 위한 사용자 인식 컴포넌트 : 영상기반 사용자 인식(얼굴 인식)과 음성기반 사용자 인식(화자 인식)에 필요한 데이터 형과 컴포넌트를 구성하기 위해 필요한 인터페이스에 대한 규격임
- URC 로봇을 위한 음원 추적 컴포넌트 : 사용자가 로봇을 호출할 때 소리를 통해 사용자가 어디에 있는지를 알아내기 위한 컴포넌트 인터페이스에 대한 규격을 정의함
- URC 로봇을 위한 사용자 추적 컴포넌트 : 영상을 통해 사용자의 위치 정보를 알아내기 위한 컴포넌트 인터페이스에 대한 규격임
- URC 로봇을 위한 자율 주행 컴포넌트 : URC 로봇의 주행에 필요한 컴포넌트 인터페이스에 대한 규격을 정의함

정부에서는 RUPI 2.0 규격에 대한 규격 보완 및 참조 구현 시스템 개발 과제를 2008년부터 시작할 예정이다. 개발된 RUPI 기술은 원칙적으로 오픈 소스(Open Source) 정책을 추진하며 교육 센터를 통해 현 로봇 개발자 및 잠재적 로봇 개발자에 대한 교육을 실시할 계획으로 로봇 산업의 저변 확대에 크게 기여할 것으로 기대된다.

3. 로봇 소프트웨어 플랫폼 연구 동향

네트워크 로봇에 대한 연구 개발이 활성화됨에 따라 로봇 소프트웨어 개발을 효과적으로 지원하기 위한 로봇 소프트웨어 플랫폼에 대한 연구도 활발해지고 있다. 본 장에서는 로봇 소프트웨어 플랫폼과 관련된 연구 동향에 대해 기술한다.

3.1 MSRS

미국 마이크로소프트사의 MSRS(Microsoft Robotics Studio)[8]는 .NET 기반의 시각적 로봇 응용 개발 환경 및 실행 환경을 지원하고 있다.

MSRS상에서 개발되는 로봇 응용은 기본적으로 서비스의 조합으로 이루어지며, 멀티 쓰레딩 환경에서 로봇 프로그래밍을 작성할 때 발생하는 동시성의 문제를 해결하고 단순한 코딩 작업을 가능하게 하는 동시성 처리 기능을 제공한다. MSRS상에서 개발된 로봇 응용은 여러 컴퓨터에 분산되어 실행 가능하며, 이들 간의 통신은 REST(REpresentational State Transfer)를 통해 이루어진다. MSRS는 Agea사의 PhysX엔진을 기반으로 한 3D 동역학 로봇 시뮬레이션(simulation) 환경과 로봇 프로그램 초보자들도 쉽게 로봇 응용 개발이 가능한 비쥬얼 개발 환경을 제공한다.

3.2 ERSP

미국 Evolution Robotics사의 ERSP(Evolution Robotics Software Platform)[9]는 로봇 응용 개발을 위한 소프트웨어 플랫폼으로서, 개발자들이 로봇 응용을 신속하고 용이하게 구축할 수 있게 하는 소프트웨어 구조, 핵심 컴포넌트 및 개발 도구를 제공한다. ERSP는 장치 추상화 계층, 행위 실행 계층, 태스크 실행 계층 등 3계층으로 이루어진 실행 환경과 물체 인식, 자율 주행, 위치 인식 등의 로봇 응용 컴포넌트 그리고 로봇 행위 개발 도구를 지원한다.

3.3 RTC

RTC(Robot Technology Component)[10]는 일본의 AIST 및 미국의 RTI 주도로 OMG 의 Robotics DTF에서 2006년 9월 표준으로 채택된 로봇 소프트웨어 컴포넌

트 규격이다.

RTC는 재사용이 가능한 로봇 소프트웨어 컴포넌트에 대한 규격으로 로봇 컴포넌트에 대한 포괄적인 내용을 정의하고 있으며, 경량 RTC(Lightweight RTC), 실행 시멘틱스(Execution Semantics), 인트로스펙션(Introspection)으로 구성되어 있다. 경량 RTC는 컴포넌트, 컴포넌트의 속성을 나타내는 프로퍼티(Property), 컴포넌트 간의 통신을 위한 포트(Port) 등을 정의하고 있다. 실행 시멘틱스는 로봇 응용에서 자주 사용되는 프로그래밍 패턴(Finite State Machine, Stimulus Response Processing 등)을 제공하여 개발자가 쉽게 로봇 응용을 개발할 수 있도록 한다. 인트로스펙션은 컴포넌트, 포트 등의 상태를 실행 시간에 검사할 수 있는 모니터링 인터페이스를 제공한다.

3.4 OROCOS

OROCOS(Open Robot Control Software)[11]는 유럽에서 2000년 12월에 로봇 제어 소프트웨어 개발을 위한 오픈 소스 프로젝트로 시작되었다. OROCOS는 실시간 컴포넌트가 가져야 할 컴포넌트 모델 및 인터페이스를 정의하고 있으며, 손쉽게 실시간 로봇 응용을 개발할 수 있도록 C++ 클래스로 이루어진 실시간 툴킷(Realtime Toolkit)과 스크립트 언어를 제공한다.

위에서 소개한 로봇 소프트웨어 플랫폼들은 로봇 응용을 손쉽게 개발하기 위해 나름대로의 기능을 제공하고 있으나, 로봇 서버 및 로봇 클라이언트를 포함한 네트워크 로봇의 모든 분야를 포괄하지는 못하고 있다.

다음 장에서는 RUPI 규격 중 로봇 내부 태스크 및 콘텐츠 실행 미들웨어의 참조 구현 시스템인 uROSE (urc RObot Service development Environment)에 대해 기술한다.

4. uROSE

1980년대 중반 MIT의 Rodney Brooks가 포섭구조(Subsumption architecture)[12]를 발표하여 행위기반 로보틱스의 새로운 장을 열었다. 행위기반 로보틱스는 종래의 SPA(Sense-Plan-Act) 구조의 단점이라고 할 수 있는 느린 응답성을 극복하고자 하였다[13]. 행위기반 로보틱스의 주요 특징은 행동지향 센싱(Action-oriented sensing), 행위의 병렬 수행, 그리고 표현(representation)의 제한적 사용이라고 볼 수 있다. 기존의 접근방식이 범용적 인지를 목적으로 한 것이라면 행동지향 센싱은 각 행동을 수행하기 위해 필요한 정도의 인지만을 수행하는 것으로서, 각 행동 별로 최적화된 인지 알고리즘이 결부된다. 이 방식의 장점은 인지과정이 빠르고 상대적으로 간결하다는 장점이 있다. 표현에 있어서 이전의 접근방식이 전역적인 표현을 사용하였지만, 행위기반 로보틱스에서는 표현의 사용을 극도로 자제하거나 최소한의 지역적인 표현만을 사용하는 것을 특징으로 한다[12]. 행위기반 로보틱스적 접근방식은 곤충수준의 지능을 표현할 수 있다는 점을 증명하였으나, 그 이상의 복잡한 태스크를 수행할 수 있다는 점을 증명하지는 못했다. 따라서 이후의 연구결과들은 전통적인 SPA의 계층적 구조의 장점과 행위기반 로보틱스의 장점을 적절히 조합한 하이브리드형 제어구조가 주류를 이루게 되었다.

uROSE는 구조의 모듈화를 꾀하고 빠른 응답성 및 상위 수준의 지능을 부여하기 위하여 하이브리드 구조를 제공하며, 기존 하이브리드 구조와 다른 점은 기존의 하이브리드 구조가 주로 단품로봇 위주의 제어구조이고 비분산화된 환경에서의 동작을 가정하고 만들었다면[14-16], uROSE는 분산 환경과 네트워크 환경을 고려하고 이식성 및 재사용성을 높이기 위해 로봇 컴포넌트의 분산화를 지향하고 있다는 점이다. 이는 하드웨어 또는 네트워크의 상황에 따라서 컴포넌트들을 적절히 분산시킬 수 있으며, 다양한 형태의 하드웨어에 적용이 가능하고 이식성을 높일 수 있다는 장점을 지니고 있다.

4.1 uROSE 계층 구조

그림 2는 uROSE의 계층 구조를 나타낸 것이다.

uROSE는 크게 태스크 플래닝 계층(Task Planning Layer), 태스크 실행 계층(Task Execution Layer), 행위 실행 계층(Behavior Execution Layer), 디바이스 추상화 계층(Device Abstraction Layer)의 4계층과 부가적으로 서비스 계층(Service Layer), 시스템 계층(System Layer)의 2계층으로 구성된다.

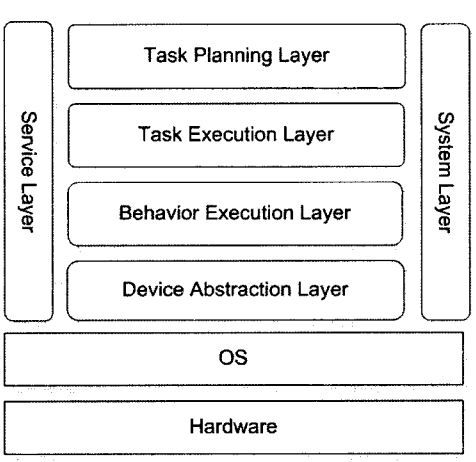

그림 2 uROSE 계층 구조

태스크 플래닝 계층은 로봇에게 주어진 상황에 대해서 태스크를 어떤 방식으로 구동시킬 것인가를 결정하거나, 복수개의 태스크가 동시 또는 순차적으로 주어졌을 때 어떤 순서로 태스크를 수행할 것인가를 결정하는 계층이다.

태스크 실행 계층은 태스크 플래닝 계층에 의해서 선택된 태스크의 실행 가능 조건 검사, 태스크의 실행, 실행된 태스크의 성공과 실패 검사, 계층적 태스크에 대한 실행 조건 검사의 기능을 수행한다. 태스크 실행 계층에서는 XML로 기술된 태스크를 객체화하고 이를 실행하는 기능을 포함하고 있다. 태스크들은 순차성, 병렬성, 동시성, 반복성의 특징을 가질 수 있으며, 태스크를 내포할 수 있다.

행위 실행 계층은 컴포넌트 형태로 개발된 행위들을 구동시키는 계층으로써 비교적 실행 시간이 짧고 빠른 응답시간이 요구되는 계층이다. 이 계층은 행위 컴포넌트의 실행 및 실행 상태 관리를 담당한다.

특정한 로봇 플랫폼을 타깃으로 개발된 응용 프로그램은 다른 플랫폼에서는 전혀 동작되지 않으며, 대개의 경우 하드웨어 의존적인 부분이 프로그램 곳곳에 산재되어 이식 작업조차 매우 곤란한 것이 현실이다. 이는 결과적으로 로봇의 행위와 응용 프로그램들을 중복 개발하게 하여 로봇기술의 발전을 저해하는 중요한 요인이 되고 있다. 디바이스 추상화 계층의 목적은 로봇에 탑재될 각종 센서 및 액추에이터들에 대한 추상화된 하드웨어 인터페이스를 제공하고 이를 실제 디바이스에 연결할 수 있는 구조를 제공하는 것이다.

uROSE는 모듈화, 이식성, 분산성을 지향하고 있다. 또한 개발 대상의 특성에 따라 상위 계층(태스크 플래닝 계층, 태스크 실행 계층)을 위한 XML기반의 태스크 기술 언어와 하위 계층(행위 실행 계층, 디바이스 추상화 계층)을 위한 컴포넌트 기반 개발 방법론을 취하고 있다[17].

4.2 uROSE 시스템 구조

그림 3은 uROSE의 시스템 구조를 나타낸 것이다.

uROSE 시스템은 계층 관리자 위에 각 모듈들이 구성되어 있으며, 각 모듈의 실제 구현은 개발자에 의해 수정될 수 있다. 즉 고정된 형태의 로직만을 제공하는 것이 아니라 시스템 관리자가 주요 모듈의 로직을 플러그인 할 수 있다. uROSE는 각 모듈간의 단일화된 내부 통로를 위해 모듈 간 통신 구조를 제공하는 계층 관리자와 이벤트 관리자(Event Manager)를 두고 있다. uROSE와 외부 시스템과의 통신은 전용 미들웨어인 DeCOF(Distributed event based Communication Framework)를 통해 이루어진다.

uROSE의 주요 모듈들의 기능을 설명하면 아래와 같다.

o 시스템 관리자(System Manager)

시스템 관리자는 uROSE 모듈의 시작, 정지, 재시작 등을 담당한다. 또한 초기 uROSE 설정 정보를 읽어서 시스템을 부팅하는 기능을 수행한다. 시스템 관리자는 각 모듈을 부팅하고 나서 최종적으로 로봇의

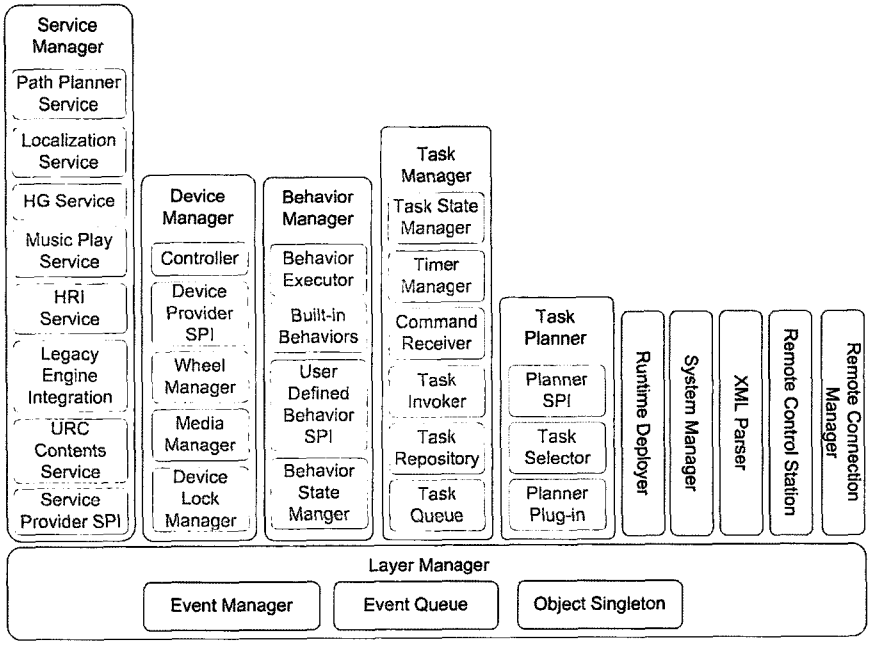

그림 3 uROSE 시스템 구조

기본 Behavior인 Startup Behavior를 수행한다. Robot Startup Behavior는 시스템 부팅 끝을 알리고 사용자에게 인사를 하는 내용의 행위이다. 이 Behavior는 시스템 설정 파일에 지정할 수 있으며 사용자가 임의로 작성하여 등록할 수 있다.

o 계층 관리자(Layer Manager)

계층 관리자는 uROSE의 모듈 간 통신 수단을 제공하기 위한 모듈이다. 시스템 관리자에 의해 생성되고 시작된 모든 모듈들의 인스턴스는 계층 관리자에 저장된다. 행위 개발자들은 계층 관리자 안의 이벤트 관리자의 이벤트(Event)를 통해서 모듈 간 또는 서비스로부터 이벤트를 받을 수 있다. 이벤트 타입(Event Type)은 사용자가 임의로 설정할 수 있다. 현재 uROSE 시스템의 서비스가 제공하는 이벤트 타입은 "asr, human-detection, location"이다.

o 서비스 관리자(Service Manager)

서비스 관리자는 system.xml에 정의된 서비스들을 동적으로 로딩하고 관리하는 역할을 수행한다. uROSE가 지원하는 시스템 서비스는 현재까지 다음과 같다.

- PathPlanner : A* 또는 다른 알고리즘으로 로봇의 주행 경로를 계획하는 서비스
- HRI 서비스 : HRI 사용자 인식 서버와 연동을 위한 프락시(Proxy)
- Home gateway Integration : Zigbee와 PLC를 통한 Home Appliance 연동서비스
- CamCapture 서비스 : 로봇 카메라로부터 영상을 JPEG으로 캡춰하여 원격지에 전송하는 서비스
- Localization 서비스 : 로봇 Localization 서버와 연동을 위한 프락시
- Robot Status 서비스 : 로봇의 상태 정보를 주기적으로 원격지에 전송하기 위한 서비스
- MusicPlay 서비스 : MP3, Wave, AU, AIFF의 음악 파일을 플레이 하기 위한 서비스
- ASR/TTS : 음성 합성과 음성 인식 서버와 연동을 위한 서비스

o 디바이스 관리자(Device Manager)

디바이스 관리자는 디바이스 제어 객체를 얻어오기 위한 관리자이며, 디바이스를 사용하기 위한 별도의 설정 XML을 로딩하고 XML에 지정된 디바이스를 서비스하기 위한 준비를 수행한다.

디바이스 관리자의 제어 대상은 두 가지 형태가 있다. 첫 번째는 로봇의 모터를 제어하거나 센서로부터 데이터를 얻어오는 휠(Wheel)과 두 번째는 음성합성, 음악 재생, 음성 및 영상의 캡춰 기능을 하는 미디어(Media)이다. 상기와 같이 두 가지 형태로 제어 대상을 구분하여 관리하는 이유는 휠은 모터와 센서들을 제어하는 별도의 하드웨어 보드와 USB, IEEE 1394, RS-232C 등을 통해 통신하는 모듈이며, 미디어는 이러한 별도의 하드웨어와 무관하게 동작하는 모듈이기 때문이다. 즉, 하위 하드웨어와의 통신 방식이 다르기 때문에 관리 객체를 별도로 두고 있다. 휠과 미디어 객체 모두 로봇 개발자가 작성하여 로봇에 플러그인할 수 있다.

디바이스 관리자의 중요 기능 중에 하나는 상위 행위 컴포넌트가 사용하는 디바이스에 대해 락킹(Locking) 기능을 제공하는 것이다. 예를 들어 어떤 행위 컴포넌트가 이동을 위해서 바퀴를 사용하고 있다면 다른 행위 컴포넌트는 바퀴를 사용하면 안되므로 이를 위해서 디바이스 관리자는 해당 리소스(Resource)에 대해서 락킹을 통해 객체의 사용을 제한하게 된다.

o 행위 관리자(Behavior Manager)

행위 관리자는 개발자에 의해서 작성된 행위(Behavior)들을 로딩하고 생명주기를 관리하는 기능을 수행한다. 행위 관리자는 태스크 관리자에 의해서 요구된 행위를 실행하는 것이 주요 기능이다. 또한 행위 관리자는 요구된 행위를 실행하고 실행 전과 후에 상태 검사를 위해서 래퍼(Wrapper) 객체를 통해 행위를 호출하도록 한다. 행위 관리자의 주요기능은 다음과 같다.

- 행위 객체 생명주기 및 실행 상태 관리
- 객체간 동기화 기능 제공
- 객체를 위한 타이머(Timer) 기능 제공
- 객체를 위한 이벤트 기능 제공
- 개발을 위한 추상화 객체 제공

o 태스크 관리자(Task Manager)

태스크 관리자는 태스크 플래너로부터 요청된 태스크의 실행을 담당하는 모듈로써, XML로 표현된 태스크를 객체화하고 태스크의 실행 형태에 따라 태스크를 실행한다. 태스크의 호출 유형은 명령, 이벤트, 시간의 세가지로 구분된다. 태스크 관리자는 태스크들의 실행준비를 하고 해당 태스크가 선택되었을 때, 태스크 플래너에게 실행 계획을 요청한다. 태스크 플래너로 부터 실행 명령을 받은 태스크들은 태스크의 실행 유형에 따라서 태스크 관리자가 실행을 하게 된다.

그림 4는 uROSE에서 제공하는 태스크 프로그래밍 모델을 나타낸 것이다.

Pre-Condition은 외부로부터 태스크가 활성화 될 때 자신의 실행 조건을 사전에 검사하는 부분이다. 검사결과 조건을 만족하면 Behavior Set을 실행한다.

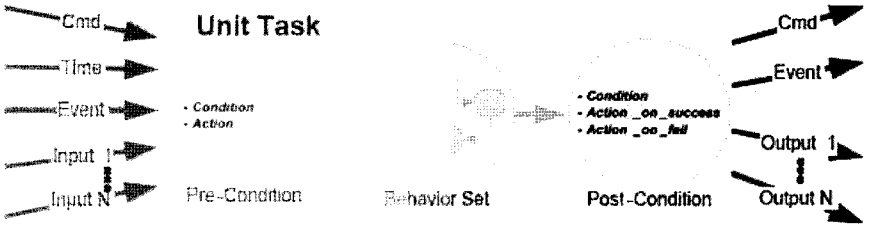

그림 4 uROSE 태스크 프로그래밍 모델

Behavior Set은 태스크의 목표(Goal)를 위해 선택된 행위들의 집합이며 순차적인 실행 특성을 갖는다. Post-Condition은 태스크가 실행을 완료한 후에 자신의 실행 성공 여부를 판단하는 조건이며, 부가적으로 하위 태스크(Nested Task)를 실행시키는 조건을 담고 있다. 또한 태스크의 성공 여부에 따라 각기 다른 액션(Action)을 수행할 수 있다. uROSE는 그림 4와 같이 모델링된 N개의 태스크를 가질 수 있으며, 외부로부터 실행에 대한 입력을 받을 수 있다. 또한 각 태스크는 자신의 상태를 기반으로 다른 태스크를 순차적으로 혹은 병렬적으로 실행할 수 있다.

o 태스크 플래너(Task Planner)

태스크 플래너는 태스크의 실행을 계획하고 태스크 관리자에게 실행을 요청하는 모듈이다. 태스크 플래너의 플래닝(Planning) 알고리즘은 서비스의 유형 및 적용 영역에 따라 달라질 수 있다. 따라서 uROSE에서는 태스크 플래너를 개발자가 작성하여 시스템에 플러그인 시킬 수 있는 구조를 지원한다. uROSE에 기본으로 탑재한 태스크 플래너의 기능은 다음과 같다.

- 태스크 객체의 실행 유형 별 실행 계획
- Planned Task의 경우Planned Task Runner의 Queue에 저장된 태스크들과 실행 순서 재조정
- 태스크 관리자에 선택된 태스크의 실행 요청

4.3 uROSE 실험

그림 5는 uROSE 시스템이 구동되고 있는 웨버 R3 로봇을 나타낸 것이다. 웨버 R3는 카메라, 초음파 센서 등이 탑재된 모바일 서비스 로봇으로 소프트웨어 로봇과 연동하여 로봇을 구동하거나, URC 서버를 통해 음성 합성, 음성 인식을 수행하고 뉴스/날씨와 같은 URC 콘텐츠 서비스를 제공할 수 있다.

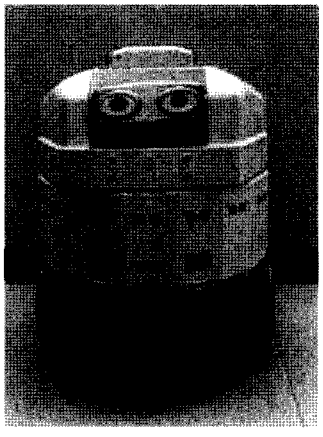

그림 5 웨버 R3

그림 6은 uROSE 시스템을 테스트하기 위한 테스트 환경을 나타낸 것으로 일반 가정환경과 유사하게 구성하였다.

그림 7은 uROSE 시스템을 이용한 간단한 테스트 시나리오를 나타낸 것이다. 테스트 시나리오는 아침 정해진 시간에 아빠를 깨우고 그날의 주요 뉴스를 읽어 주는 것이다.

위에서 기술한 테스트 시나리오는 그림 8의 태스크 저작도구에 의해 uROSE에서 실행 가능한 태스크로 저작되어 XML 파일로 저장된다.

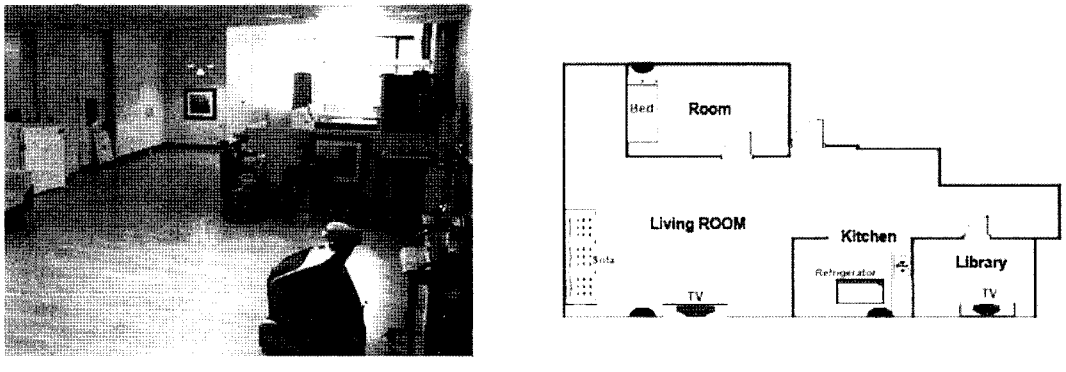

그림 6 테스트 환경

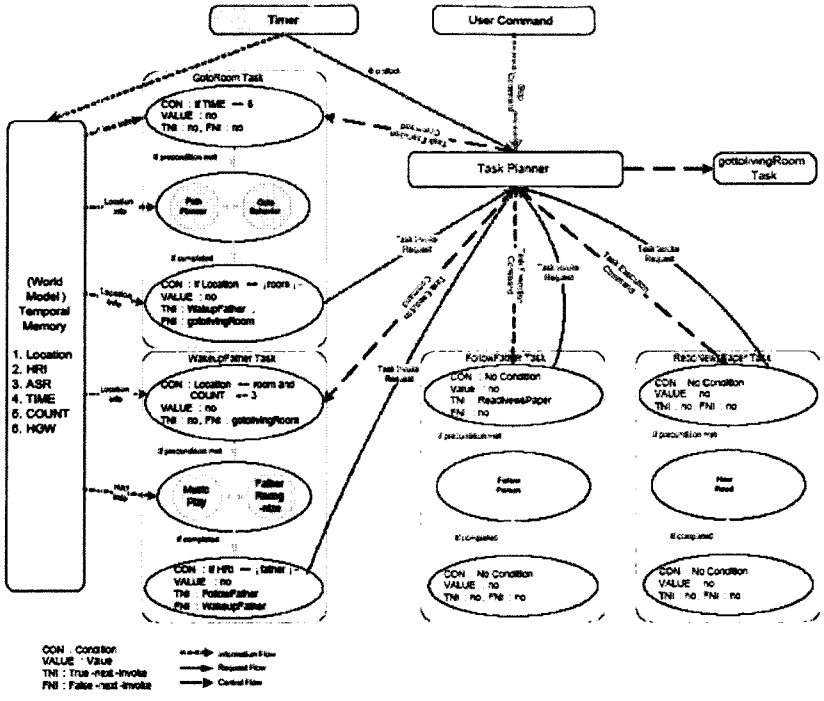

그림 7 테스트 시나리오

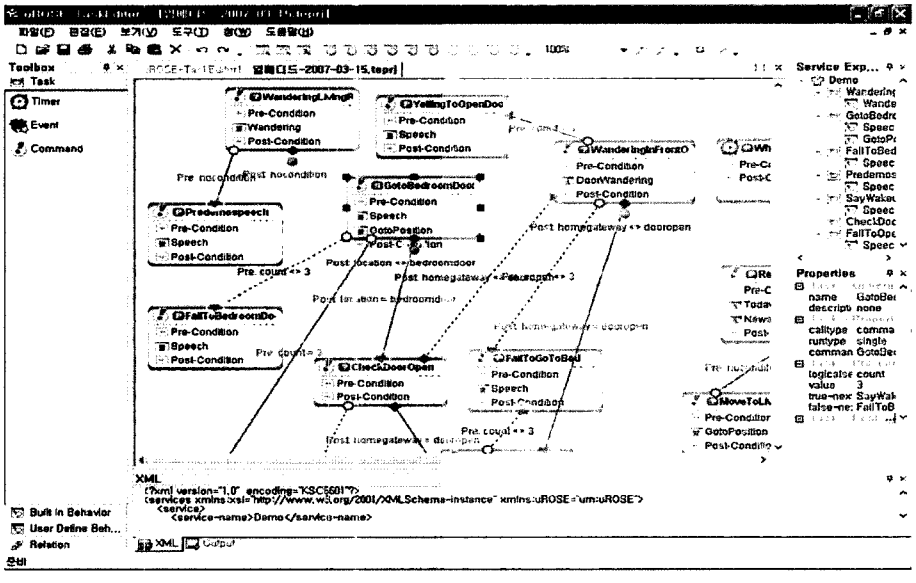

그림 8 태스크 저작도구

XML 파일로 저장된 태스크는 그림 9에 나타난 태스크 배포 및 모니터링 도구에 의해 uROSE 시스템에 배포되어 실행되게 된다. 그림 9의 도구를 통해 태스크의 배포, 실행 시작 및 중지, 태스크의 실행 상태 모니터링, 가전기기 제어, 로봇 영상 디스플레이, 로봇 상태 모니터링 등이 가능하다.

5. 결 론

본 고에서는 RUPI-Client 참조 구현인 uROSE 시스템에 대해 기술하였다. 네트워크 로봇은 유비쿼터스 환경과 연계하여 다양한 정보를 획득하고 사용자에게 능동적인 서비스를 제공할 수 있게 된다. 다양한 형태의 네트워크 로봇이 유비쿼터스 환경에서 사용자에게 보다 능동적이며 풍부한 서비스를 제공하기 위해서는 uROSE와 같은 표준화된 형태의 로봇 플랫폼이 필수적이라고 할 수 있다.

현재 RUPI 2.0 규격이 공개되었으며 향후 RUPI 규격은 로봇 업체를 비롯한 다양한 기관의 요구사항을

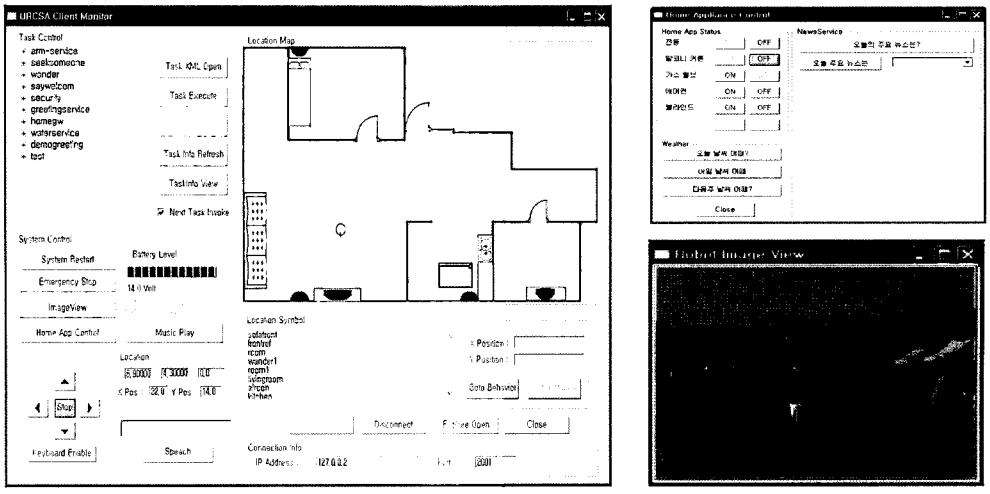

그림 9 uROSE 실행 화면

수용하여 더욱 발전할 것으로 예상되며, uROSE도 그에 따라 좀 더 편리하고 다양한 기능을 제공해야 할 것이다.

참고문헌

[1] 김성훈, 김중배, "URC를 위한 로봇 S/W 아키텍처 기술", 대한전자공학회 특집호, 제33권, 제3호, pp. 56~63, 2006.

[2] 김혜진, 이재연, "지능형 서비스 로봇으로의 이동", 정보통신연구진흥원 주간기술동향, 통권 1181호, pp. 26~34, 2005.

[3] 손주찬, 김재홍, 하영국, 장민수, 박천수, 이미경, "URC 지능형 서비스 기술 개발 동향", 정보통신연구진흥원 주간기술동향, 통권 1190호, pp.30~44, 2005.

[4] 김현, 이강우, 이주행, 강태근, 문애경, 서영호, 조준면, "URC에서의 소프트웨어 로봇 기술", 한국통신학회지, 제21권, 제10호, pp.36~43, 2004.

[5] 김종환, 이강희, 김태훈, "Sobot, Embot과 Mobot으로 구성된 유비쿼터스 로봇", 제어 · 자동화 · 시스템공학회지, 제11권, 제4호, pp.45~53, 2005.

[6] Ha Y. G., Sohn J. C., Cho Y. J., Yoon H. S., "Towards ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework," ETRI Journal, Vol. 27, No. 6, pp.666~676, Dec. 2005.

[7] RUPI, http://www.rupi.or.kr

[8] MSRS, http://www.microsoft.com/korea/robotics/

[9] ERSP, http://www.evolution.com/

[10] "The Robotic Technology Component Specification," OMG Adopted Specification, ptc/06-11-07, Nov. 2006.

[11] OROCOS, http://www.orocos.org/

[12] R. A. Brooks, "Intelligence without representation," Artificial Intelligence, Vol. 47, pp.139~159, 1991.

[13] Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P. and Slack, M. G., "Experiences with an architecture for intelligent reactive agents," Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9, No. 2, pp.1237~1256, 1997.

[14] Connell, J., "SSS: A hybrid architecture applied to robot navigation," Proceedings of the IEEE Conference on Robotics and Automation(ICRA), pp.2719~2724, 1992.

[15] Gat, E., "Reliable goal-directed reactive control for real-world autonomous mobile robots," Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1991.

[16] Bonasso, R. P., "Integrating reaction plans and layered competences through synchronous control," Proceedings of the International Joint Conference on Artificial Intelligence, pp.1225~1231, 1991.

[17] 이승익, 장철수, 정승욱, 김중배, "로봇 소프트웨어 아키텍처의 연구동향과 현황", 정보통신동향분석, 제20권, 제2호, pp.1~13, 2005.

정승욱

1996 전남대학교 학사
1998 광주과학기술원 석사
1998~현재 한국전자통신연구 선임연구원
관심분야 : 로봇 소프트웨어 아키텍처, 컴포넌트 기반 로봇 응용 개발, 지능형 서비스 로봇, 미들웨어
E-mail : swjung@etri.re.kr

이승익

1995 연세대학교 학사
1997 연세대학교 석사
2002 연세대학교 박사
2002 일본 ATR 연구소 연구원
2004~현재 한국전자통신연구원 선임연구원
관심분야 : 로봇 소프트웨어 아키텍처, 컴포넌트 기반 로봇 응용 개발, 지능형 시스템, 복잡성 과학
E-mail : the_silee@etri.re.kr

김성훈

1995 광운대학교 학사
1997 광운대학교 석사
2007 한양대학교 박사과정
1996~현재 한국전자통신연구원 선임연구원
관심분야 : 로봇 소프트웨어 아키텍처, 컴포넌트 기반 로봇 응용 개발, 지능형 서비스 로봇, 미들웨어
E-mail : saint@etri.re.kr