상황인지 워크플로우와 서비스 지향 미들웨어를 이용한 URC 로봇 소프트웨어 아키텍처

A Software Architecture for URC Robots using a Context-Aware Workflow and a Service-Oriented Middleware

곽 동 규[1], 최 종 선[1], 최 재 영†, 유 재 우[2]

Dong-Gyu Kwak1, Jongsun Choi1, Jaeyoung Choi†, Chae-Woo Yoo[2]

Abstract A URC, which is a Ubiquitous Robot Companion, provides services to users in ubiquitous computing environments and has advantage of simplifying robot's hardware and software by distributing the complicated functionality of robots to other system. In this paper, we propose SOWL, which is a software architecture for URC robots and a mixed word of SOMAR and CAWL. SOWL keeps the advantages of URC and it also has the loosely-coupled characteristics. Moreover it makes it easy to develop of URC robot software. The proposed architecture is composed of 4 layers: device software, robot software, robot application, and end user layer. Developers of the each layer is able to build software suitable for their requirements by combining software modules in the lower layer. SOWL consists of SOMAR and CAWL engine. SOMAR, which is a middleware for the execution of device software and robot software, is based on service-oriented architecture(SOA) for robot software. CAWL engine is a system to process CAWL which is a context-aware workflow language. SOWL is able to provide a layered architecture for the execution of a robot software. It also makes it possible for developers of the each layer to build module-based robot software.

Keywords : Ubiquitous Computing, URC, Software Architecture, SOWL, SOMAR, Service-Oriented Architecture(SOA), CAWL, Context-aware Workflow

1. 서 론

지능형 서비스 로봇은 다양한 환경에서 네트워크 시스템과 연계하여 인간과 상호작용하며, 상황에 따라 주어진 역할을 수행한다[1]. 유비쿼터스 컴퓨팅 환경에서 동작하는 네트워크 기반의 URC (Ubiquitous Robotic Companion) 로봇은 네트워크를 통해 외부 소프트웨어와 상호 연동하여 동작할 수 있는 구조를 갖는다[2]. 이와 같은 구조는 로봇이 수행해야 하는 기능 중에 일부를 외부 소프트웨어로 분담시켜 로봇 하드웨어나 소프트웨어의 구성을 단순화시키는 효과를 가진다. 또한 로봇 클라이언트의 원가를 절감시키고, 로봇의 서비스가 외부 소프트웨어에 다른 서비스를 요청할 수 있는 다양한 서비스 환경을 구축할 수 있다.

일반적으로 유비쿼터스 컴퓨팅 환경에서 로봇 소프트웨어 개발 환경은 세 개의 개발자 계층[3]과 하나의 사용자 계층을 가진다. 첫 번째 개발자 계층은 로봇 디바이스 소프트웨어의 개발자가 속하는 계층이고, 두 번째 개발자 계층은 로봇 디바이스를 조합하여 로봇을 개발하는 로봇 소프트웨어 개발자 계층이며, 세 번째 개발자 계층은 로봇을 실제 유비쿼터스 컴퓨팅 환경에 적용하는 로봇 응용 개발자 계층이다. 그리고 사용자 계층은 로봇 응용을 이용하여 사용자 자신의 스케줄에 따라 로봇을 동작시킨다.

Received : Apr. 1. 2010; Reviewed : Jul. 21. 2010; Accepted : Aug. 13. 2010
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원 사업의 연구결과로 수행되었음" (NIPA-2009-(C1090-0902- 0007))
† 교신저자 : 숭실대학교 컴퓨터학부 교수(choi@ssu.ac.kr)
[1] 숭실대학교 컴퓨터학부 박사과정(coolman@ss.ssu.ac.kr, jschoi@ss.ssu.ac.kr)
[2] 숭실대학교 컴퓨터학부 교수(cwyoo@ssu.ac.kr)

각 개발자는 하위 개발자 계층에서 작성한 소프트웨어나 로봇 응용의 조합으로 소프트웨어나 로봇 응용을 개발한다. 그러므로 소프트웨어의 효과적인 개발을 위해 개발자 계층의 환경에 따른 개발 계층이 요구된다. 그림 1은 각 개발자 계층을 보여주고 있다.

다수의 개발자와 사용자를 위한 유비쿼터스 컴퓨팅 환경의 로봇을 위한 소프트웨어 구조에 관한 연구는 세 가지로 분류할 수 있다. 첫 번째는 유비쿼터스 컴퓨팅 환경에 로봇을 적용하기 위한 연구로 상황인지 기반 로봇 시스템이다[4,5]. 그리고 두 번째는 다수의 개발자가 효율적으로 소프트웨어를 개발하기 위해 서비스 지향 구조를 로봇 소프트웨어에 적용하는 연구이다[6-8]. 세 번째는 각기 다른 사용자 스케줄을 고려하기 위해 워크플로우를 로봇 소프트웨어에 적용하는 연구이다[9,10]. 하지만 이 연구들은 각 요구사항을 충족하고 있지만, 로봇 개발 환경의 개발자 계층은 고려하고 있지 않아 각 개발자 계층에 효율적으로 적용하기 어렵다. 다수의 개발자가 존재하는 소프트웨어 개발 환경에서는 소프트웨어의 결합도 낮추어야 하고 이를 위해서 서비스 지향적인 개발 방법에 대한 연구가 진행되었다[6].

본 논문에서는 지능형 서비스 로봇의 소프트웨어를 효과적으로 개발하기 위해, 개발자 계층의 특성을 고려한 상황인지 기반 서비스 지향 URC 로봇 소프트웨어 아키텍처인 SOWL (SOMAR and CAWL)을 제안한다. SOWL은 서비스 지향적인 로봇 소프트웨어 미들웨어인 SOMAR (Service Oriented Middleware Architecture for Robot)[11,12]를 이용하여 다수의 개발자 계층에서 작성한 소프트웨어의 모듈화를 높인다. 또한, SOWL 아키텍처는 상황인지 기반의 워크플로우 언어인 CAWL (Context-Aware Workflow Language)[13]을 이용하여 상황정보에 따른 서비스를 사용자에게 제공할 수 있도록 구성되어 있다.

SOMAR는 로봇 디바이스 소프트웨어 개발자와 로봇 소프트웨어 개발자가 개발한 소프트웨어가 동작하는 환경으로 실제 로봇에서 로봇을 제어하는 소프트웨어가 실행되는 로봇 클라이언트와 로봇 응용 소프트웨어가 동작하는 로봇 응용 서버로 이루어진다. 로봇 응용 소프트웨어는 로봇에서 수행해야 하는 기능 중에 일부를 수행하는 소프트웨어로서 소프트웨어를 분담시켜 로봇 하드웨어나 소프트웨어의 구성을 단순화시킨다.

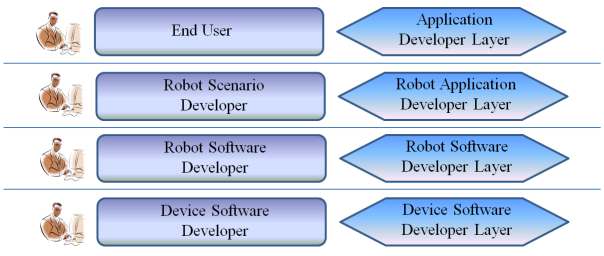


그림 1. 로봇 개발환경의 개발자 계층

CAWL은 로봇 응용 개발자와 사용자가 작성하기 위한 상황인지 기반의 워크플로우 언어로서, 사용자의 상황에 따라 서비스를 호출하거나 워크플로우 흐름의 분기를 제어할 수 있는 방법을 제공한다. 로봇 응용 개발자는 서브 워크플로우(subworkflow) 형태로 로봇이 유비쿼터스 컴퓨팅 환경에서 제공할 수 있는 서비스의 흐름을 작성한다. 서브 워크플로우는 다른 워크 플로우에서 호출할 수 있는 워크플로우이다. 그리고 사용자는 로봇 응용 개발자가 작성한 서브 워크플로우를 이용하여 자신의 스케줄에 맞는 로봇 서비스 워크플로우를 작성한다.

SOMAR와 상황인지 기반 워크플로우 시스템으로 구성되어 있는 SOWL은 다수의 개발자 계층을 갖는 로봇 개발 환경에서 각 개발자 계층이 사용하는 소프트웨어 계층을 나누어 소프트웨어의 낮은 결합도를 보장하고, XML을 이용한 로봇 데이터(로봇 제어 명령과 센싱 정보) 관리로 높은 응집도를 갖는다. 그리고 상황인지 기반의 워크플로우 언어를 사용하여 유비쿼터스 컴퓨팅 환경에 적용하기 용이하며, 사용자는 로봇 응용 개발자가 작성한 서브 워크플로우를 이용하여 손쉽게 자신의 스케줄에 따른 로봇 서비스를 제공받을 수 있다.

본 논문은 2장에서 관련 연구를 보이고, 3장에서 SOWL의 구조에 관해 논하고 4장에서 결론을 맺는다.

2. 관련 연구

본 논문의 목적은 다양한 개발자 계층이 있는 로봇 소프트웨어 환경에서 높은 모듈화를 이루고, 유비쿼터스 컴퓨팅 환경에서 적용하기 용이한 소프트웨어 아키텍처를 제안하는데 있다. 다수의 개발자와 사용자를 위한 유비쿼터스 컴퓨팅 환경의 네트워크 로봇을 위한 소프트웨어 구조에 관한 연구는 세 가지로 분류할 수 있다. 세 가지 연구는 유비쿼터스 컴퓨팅 환경에 적용하기 위한 상황인지 기반 로봇 시스템과 서비스 지향 로봇 소프트웨어, 워크플로우를 이용한 로봇 시스템이다. 본 장에서는 세 가지 연구에 관해 살펴본다.

2.1 URC 로봇과 상황인지 기반 로봇 시스템

URC 로봇은 언제 어디서나 나와 함께 하며 나에게 필요한 서비스를 제공하는 로봇을 의미한다[14]. 로봇을 유비쿼터스 컴퓨팅 환경에 적용하기 위한 연구로서 유비쿼터스 컴퓨팅 환경에 대한 요구 사항이 높아짐에 따라 URC

로봇에 관한 연구가 다양하게 진행되고 있다 [4,15]. 그 중 CAMUS (Context-Aware Middleware for URC System)[4]는 URC 로봇의 소프트웨어를 효율적으로 실행하기 위해서는 상황 정보를 지원하는 소프트웨어 미들웨어가 필요함을 보이고, URC 로봇을 위한 상황인지 기반의 소프트웨어 미들웨어를 제안하였다.

CAMUS는 로봇 소프트웨어의 목적을 세 가지로 분류하여 각 목적에 부합하도록 소프트웨어 미들웨어를 설계하였다. 첫 번째는 유비쿼터스 컴퓨팅 환경의 지원이다. 그리고 두 번째는 상황인지 기능을 보유하는 것이며, 세 번째는 지능적이고 능동적으로 사용자에게 서비스를 지원하는 것이다. CAMUS 시스템은 위의 세 가지 URC 로봇 소프트웨어 목적에 충족하도록 소프트웨어 관점에서 실제 세계를 물리공간과 가상공간으로 구분하고 모델링하였다. 그리고 상황인지를 위해 JESS (Java Expert System Shell)[16] 추론 엔진을 사용하였다. 하지만 디바이스 소프트웨어 개발자와 로봇 소프트웨어 개발자, 로봇 응용 개발자가 존재하는 로봇 소프트웨어 개발환경에 적용하기 어렵고, 각기 다를 수 있는 사용자의 스케줄을 고려하고 있지 않다.

2.2 서비스 지향 로봇과 디바이스 소프트웨어

서비스 지향 구조(SOA : Service-Oriented Architecture)는 소프트웨어 기능을 서비스로 보고, 그 서비스를 연동하여 시스템 전체를 구축해 나가는 소프트웨어 개발 방법론이다. 로봇 소프트웨어의 구조가 복잡해지고 개발자가 많아짐에 따라 결합도를 낮추고 응집도를 높이기 위해 서비스 지향 구조를 로봇 소프트웨어 개발 방법론으로 적용하는 연구가 있다[6-8]. 그 중 대표적인 연구인 SORA (Service Oriented Robotic Architecture)는 달 탐사 로봇의 소프트웨어 구조로 사용하기 위하여 개발되었다. SORA에서는 로봇 개발 환경이 다수의 개발자가 서로 다른 팀에 소속되어 많은 소스를 작성해야 하는 어려움을 지적하고, 이를 극복하기 위해 서비스 지향 구조를 로봇 소프트웨어에 적용하는 방법을 제안하였다. SORA는 서비스 지향 구조를 이용하여 소프트웨어간의 결합도를 낮추어 다수의 개발자가 존재하는 로봇 소프트웨어 개발에 서비스 지향 구조가 유용함을 보였다. 하지만 SORA는 NASA (National Aeronautics and Space Administration)와 같이 단일 조직에서 다수의 개발자를 위한 서비스 소프트웨어 인터페이스를 제시하고 있어, 각기 다른 조직에서 개발하는 개발자들에게 적용하기가 용이하지 않다. 그리고 사용자의 스케줄이나 유비쿼터스 컴퓨팅 환경을 고려하지 않았다.

또 다른 서비스 지향 구조를 이용한 로봇 소프트웨어 아키텍처 연구로는 서비스 지향 구조의 소프트웨어가 객체 지향 구조의 소프트웨어보다 능동적인 개발 프로세스를 가질 수 있음을 보이고, 이를 로봇 환경에 적용한 Yinong Chen의 연구가 있다[8]. Chen은 서비스 지향 구조가 결합도를 낮추고 응집도를 높이는 한 방법이고, 다양한 응용(XML, WSDL, OWL, BPEL 등)을 사용할 수 있음에 착안하여 서비스 지향 구조를 로봇에 적용하였다. Chen은 개발된 서비스를 서비스 중개인(Service Broker)을 통해 공개하고, GUI 기반 로봇 개발 프로그래밍 언어인 VPL (Visual Programming Language)[17]로 공개된 서비스를 조합하여 로봇 소프트웨어를 개발하였다. 개발된 로봇 소프트웨어는 실제 로봇 보드에 업로드(upload)하여 동작시킨다. 이 방법은 로봇 소프트웨어 개발 당시 최신 서비스를 업로드할 수 있는 장점을 가지나, 소프트웨어 업로드가 이루어진 후에 업데이트된 서비스에 대해서는 업데이트를 적용하기 어렵다.

2.3 워크플로우를 이용한 로봇 시스템

워크플로우는 작업 절차를 통한 정보 또는 업무의 이동을 의미한다. 사용자나 로봇 응용 개발자는 워크플로우 언어를 이용하여 작업 절차를 편리하게 기술할 수 있다. 이와 같은 워크플로우를 로봇 환경에 적용하기 위한 연구는 로봇 소프트웨어를 서비스 지향 구조로 설계하는 연구와 함께 진행되어 왔다[9,10]. 그 중 Tavetanov[9]는 워크플로우 중 많은 응용에서 사용하고 있는 BPEL[18]을 이용하여 로봇의 서비스 프로세스를 제어하는 방법을 연구하였다. Tavetanov가 제안하는 시스템은 사용자가 웹 브라우저를 통해 자신이 요구하는 로봇 응용에 따라 BPEL 편집기를 이용하여 BPEL 문서를 작성할 수 있고, 작성한 BPEL 문서에 따라 모바일 로봇을 제어할 수 있다. 하지만 유비쿼터스 컴퓨팅 환경에서 사용하기 위해서는 사용자와 로봇의 상황 정보를 적용시킬 수 있어야 하는데, BPEL은 상황 정보를 처리할 수 있는 방법을 가지고 있지 않다. BPEL과 같이 기존의 워크플로우 언어에 상황 정보를 처리하기 위한 연구[19,20]가 진행되었지만, 이는 기존 워크플로우 엔진 이외에 다른 요소를 포함하여 시스템의 구조가 복잡해지고, 기존의 문법을 수정하거나 새로운 문법을 도입해야 한다.

3. URC 로봇 소프트웨어 아키텍처 SOWL

로봇 소프트웨어 아키텍처인 SOWL의 목적은 로봇에서 실행되는 로봇 소프트웨어의 모듈화를 높이고, 상황인지 워크플로우 기술을 도입하여 유비쿼터스 컴퓨팅 환경

에 용이하게 적용시키는데 있다. 로봇 소프트웨어 개발 환경은 앞의 그림 1과 같이 세 개의 개발자 계층과 하나의 사용자 계층을 갖는다. 표 1은 각 개발자 계층의 주요 요구사항을 보여준다.

디바이스 소프트웨어 개발자는 로봇에 장치할 디바이스를 동작시키는 소프트웨어를 개발하는 개발자이다. 디바이스 소프트웨어는 디바이스가 장치될 로봇에서 동작하며 디바이스를 직접 제어해야 한다. 그리고 로봇의 하드웨어는 다수의 디바이스를 조합하여 구성하고, 각 디바이스를 제어하는 소프트웨어와 로봇의 요구사항에 따라 응용 소프트웨어가 필요하다. 로봇 소프트웨어 개발자는 로봇의 요구사항에 따른 응용 소프트웨어와 디바이스 소프트웨어로 로봇 소프트웨어를 개발한다. 즉, 로봇 소프트웨어 개발자는 단일 로봇의 요구사항을 프로그램으로 작성한다. 로봇 응용 개발자는 로봇이 적용될 유비쿼터스 컴퓨팅 환경에서 상황 정보에 따른 로봇의 서비스를 정하고, 이를 서브 워크플로우를 이용하여 작성한다. 서브 워크플로우는 한 번 작성한 워크플로우를 다른 워크플로우에 포함시킬 수 있는 기능을 갖는다. 상황 정보는 추상화된 정보이나 최종 사용자가 이해하기는 어려워 유비쿼터스 컴퓨팅 환경과 상황 정보를 잘 이해하는 로봇 응용 개발자가 작성해야 한다. 그리고 최종 사용자는 로봇 응용 개발자가 개발한 서브 워크플로우를 이용하여 자신에 스케줄에 따른 워크플로우를 작성한다.

본 논문은 각 개발자 계층에게 독립적인 개발 환경을 제공하여 결합도를 낮추기 위한 로봇 소프트웨어 아키텍처를 제안한다. 소프트웨어 계층간 낮은 결합도는 소프트웨어 개발자 간의 높은 수준의 학습을 요구하지 않는다. 제안하는 아키텍처는 디바이스 소프트웨어 개발자와 로봇 소프트웨어 개발자에게 SOMAR 미들웨어를 제공하고, 로봇 응용 개발자와 최종 사용자에게는 CAWL 워크플로우 시스템을 제공한다. 그림 2는 각 개발자 계층에게 제공되는 환경을 보인다.

그림 2와 같이 SOWL은 세 개의 실행 계층을 갖는다. 첫 번째 계층은 로봇 응용 개발자와 사용자가 작성한 서브 워크플로우나 사용자의 스케줄을 기술한 워크플로우가 실행되는 상황인지 기반 워크플로우 시스템이다. 그리고 두 번째 계층은 로봇 개발자가 작성한 로봇 응용 서비스가 실행되는 계층이다. 로봇 응용 서비스는 로봇의 요구사항에 따라 개발된 응용 소프트웨어가 서비스로 추상화된 소프트웨어로서 디바이스 소프트웨어를 호출하여 요구

표 1. 로봇 소프트웨어 개발 계층과 주요 요구사항

개발자 계층	개발 소프트웨어	주요 요구사항
최종 사용자	스케줄	자신의 스케줄에 맞는 서비스
로봇 응용 개발자	로봇 응용	로봇 서비스에 따른 상황 정보와 서비스 간 관계 정의
로봇 소프트웨어 개발자	로봇 소프트웨어	디바이스의 조합으로 로봇을 동작하는 소프트웨어
디바이스 소프트웨어 개발자	디바이스 소프트웨어	디바이스를 동작 시키는 소프트웨어

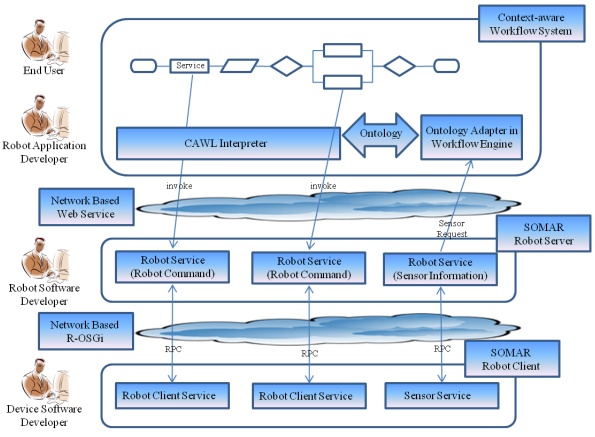


그림 2. SOWL의 실행 구조

표 2. URC 로봇 S/W 플랫폼 비교

구분	SOWL	RSCA	OPRoS	ERSP
기반 환경	R-OSGi	RT-CORBA	DMP & S/W 컴포넌트 미들웨어	ERSP
S/W 계층 구조	4 계층	3 계층	4 계층	4계층
S/W 단위	서비스	컴포넌트	컴포넌트	컴포넌트
URC 서버 지원	지원	지원	지원	지원하지 않음
RPC 지원	R-OSGi Proxy Class Web Services	RT-CORBA	OPRoS 2.0.3-R11.2 (RPC Layer)	이진 목적 소스
워크플로우 지원	CAWL	지원하지 않음	지원하지 않음	지원하지 않음

사항을 만족시킨다. 세 번째 계층은 로봇 디바이스 개발자가 작성한 로봇 클라이언트 서비스가 실행되는 계층이다. 로봇 클라이언트 서비스는 디바이스 소프트웨어가 서비스로 추상화된 소프트웨어로서 각 디바이스를 제어한다. 제안하는 로봇 소프트웨어 아키텍처는 각 개발자 계층에게 독립적인 소프트웨어 개발/실행 환경을 제공한다. 표 2는 URC 로봇 S/W 플랫폼과 제안하는 SOWL을 비교한 내용을 나타낸다[3,21-23].

3.1 로봇 소프트웨어 미들웨어 SOMAR

로봇 소프트웨어의 모듈화를 위한 SOMAR 미들웨어는 로봇 클라이언트와 로봇 응용 서버로 구분된다. 로봇 클라이언트는 디바이스가 장치되어 있는 물리적 로봇에서 실행되는 디바이스 소프트웨어의 실행 계층이다. 그리고 로봇 응용 서버는 로봇 클라이언트에서 동작할 소프트웨어 중 일부를 분담하여 실행하는 계층으로 로봇 소프트웨어가 실행된다. 이와 같은 실행 구조는 로봇 클라이언트에서 동작하는 소프트웨어의 구조를 단순화시키고, 로봇 디바이스 개발자와 로봇 소프트웨어 개발자에게 독립적인 개발 환경을 제공하여 두 계층간의 소프트웨어 모듈화를 보장한다.

물리적 로봇의 소프트웨어 계층인 SOMAR 로봇 클라이언트는 로봇 디바이스 소프트웨어가 실행된다. 로봇은 다수의 디바이스로 구성되어 있는데, 일반적인 컴퓨팅 시스템과는 다른 특성을 갖는다. 로봇에 사용되는 디바이스는 매우 다양한 기능을 가지고 있으며 디바이스 기능을 사용하기 위한 인터페이스도 다양하다. 이와 같은 로봇 디바이스의 다양성은 로봇 소프트웨어 개발을 어렵게 하는 요인이 된다. 그러므로 로봇 소프트웨어를 효율적으로 개발하기 위해서는 로봇 디바이스 소프트웨어의 추상화된 모듈화가 필요하다. 로봇 디바이스 서비스는 디바이스 소프트웨어의 추상화/모듈화한 서비스로서 로봇 클라이언트 계층에서 실행된다. 로봇 디바이스 서비스는 상위 계층에서 받은 로봇 제어 데이터를 분석하여 해당 디바이스에게 데이터를 전달하는 역할을 한다. 그림 3은 한 대의 URC 로봇 클라이언트에서 물리적 디바이스와 디바이스 소프트웨어, 로봇 디바이스 서비스의 관계를 보여준다.

디바이스 소프트웨어는 디바이스 개발자나 개발회사에서 디바이스와 함께 제공한다. SOMAR 로봇 클라이언트는 데이터 표현의 W3C 표준으로 많은 응용에서 사용하고 있어 개발자가 특별히 학습하지 않고 사용할 수 있다. 디바이스 소프트웨어 개발자는 개발한 디바이스 제어에 필요한 정보를 XML 문서로 기술할 수 있도록 디바이스 제어 XML 스키마(Schema)를 디바이스 소프트웨어와 함

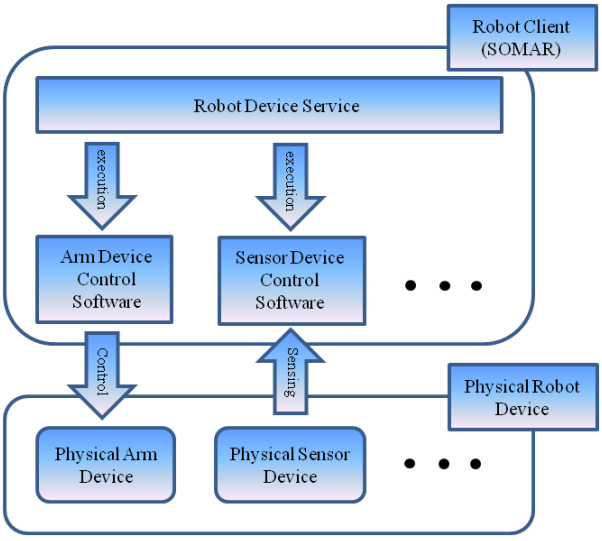


그림 3. URC 로봇 클라이언트 계층의 소프트웨어 실행 구조

께 개발한다. 그리고 로봇 개발자는 디바이스 개발자가 개발한 XML 스키마의 조합으로 로봇 제어 XML 스키마를 개발한다. 해당 디바이스 제어 데이터는 해당 디바이스 소프트웨어로 전달되어 실제 디바이스를 제어한다. 이와 같이 XML 제어 데이터를 기반으로 로봇 제어 데이터와 디바이스 제어 데이터를 작성하면 로봇 소프트웨어 개발이 용이하다.

SOMAR 로봇 응용 서버는 로봇 클라이언트에서 실행해야 하는 응용 소프트웨어 중 일부를 실행하고 웹 서비스를 기반으로 하는 워크플로우와 같은 다른 응용에서 로봇이 제공하는 서비스를 사용할 수 있도록 서비스를 중계하는 역할을 한다. 웹 서비스는 느슨한 결합을 보장하여 SOA 기반의 워크플로우 시스템[19,24,25]이나 JWS (Java Web Services)[26] 등과 같은 응용에서 많이 사용되고 있다. 그러므로 웹 서비스를 기반으로 하는 로봇 서비스는 다양한 환경에서 로봇의 기능을 제어할 수 있는 장점을 갖는다. 그림 4는 한 대의 로봇을 제어하고 있는 로봇 응용 서비스와 디바이스 서비스 간의 관계를 보여준다.

그림 4와 같이 로봇 응용 서비스는 로봇 웹 서비스와 로봇 서비스로 구성되고 로봇 서비스는 R-OSGi에서 제공하는 RPC 구조인 프록시(Proxy) 클래스를 이용하여 로봇 디바이스 서비스와 통신한다. R-OSGi 기반 프록시 클래스는 원격 컴퓨팅 환경에서 실행되는 객체를 단일 컴퓨터에서 실행되는 객체와 동일하게 프로그래밍할 수 있는 추상적인 인터페이스를 제공한다. 그리고 로봇 웹 서비스는 로봇을 이용하는 다른 응용에서 웹 서비스를 이용하여 로봇이 제공하는 서비스를 사용할 수 있도록 한다. 웹 서비스는 일반적으로 XML을 이용하여 데이터를 전송하는 경

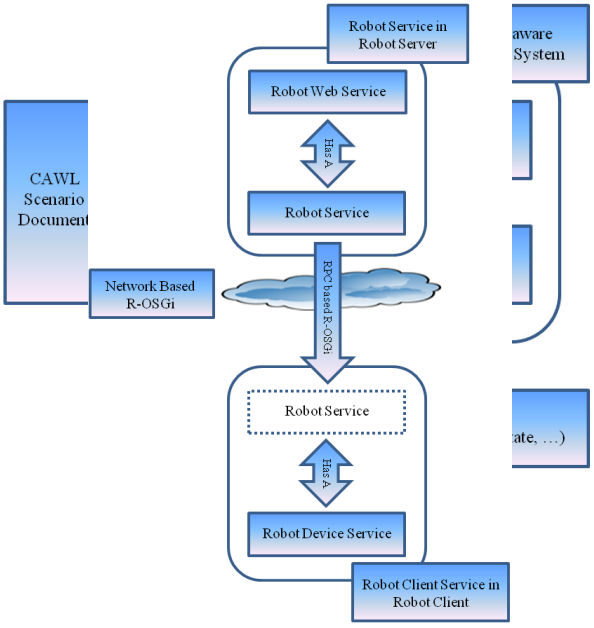


그림 4. 로봇 응용 서비스와 로봇 디바이스 서비스간의 관계

우가 많다. 그러므로 로봇 제어 XML을 이용한 로봇 웹 서비스는 개발자에게 높은 수준의 학습을 요구하지 않는다.

3.2 상황인지 기반 워크플로우 시스템

상황인지 기반 워크플로우 시스템은 로봇 응용 개발자와 사용자가 작성한 워크플로우 문서를 동작시킨다. 로봇 응용 개발자는 상황 정보에 따라 로봇 소프트웨어 개발자가 개발한 로봇 응용 서비스를 호출하는 서브 워크플로우 문서를 작성한다. 서브 워크플로우는 한 번 작성한 워크플로우 문서를 다른 워크플로우 문서에 포함시키기 위해 CAWL에서 제안된 워크플로우 문서 작성 방법이다. 상황 정보는 현실 세계를 추상화한 정보이나 최종 사용자가 학습하여 사용하기는 어렵다. 그러므로 유비쿼터스 컴퓨팅 환경과 상황 정보를 이해하는 로봇 응용 개발자가 작성해야 한다. 최종 사용자는 로봇 응용 개발자가 작성한 서브 워크플로우를 이용하여 자신의 스케줄에 맞는 워크플로우 문서를 작성할 수 있다. 상황인지 정보에 따른 서비스 호출은 로봇 응용 개발자에 의해 서브 워크플로우로 작성되어 있어 사용자는 서브 워크플로우의 호출로 자신의 스케줄에 따른 워크플로우를 쉽게 작성할 수 있다. 그림 5는 서브 워크플로우 정의와 호출의 예를 보이고 있다[13].

그림 5의 9 ~ 11줄에서 “MeetingRoomService”라는 서브 워크플로우를 정의하였고 4줄에서 정의한 서브 워크플로우를 호출하고 있다. CAWL 언어는 서브 워크플로우와 함께 다수의 사용자를 위한 다중 워크플로우 기능도 제공하고 있다. 다중 워크플로우 기능은 다수의 사용자 스케

```
1  <flow name="UserA_Flow">
2    <node name="">
3      ...
4      <invoke subflow="MeetingRoomSerivce" />
5      ...
6    </node>
7  </flow>
8  ...
9  <flow name="MeetingRoomService">
10   ...
11 </flow>
```

그림 5. 서브 워크플로우의 정의와 호출

줄이 동일한 유비쿼터스 컴퓨팅 환경에서 적용될 경우 각 사용자의 스케줄이 동적으로 적용할 수 있는 기능이다. 다중 워크플로우 기능은 다수의 사용자가 다수의 로봇을 제어하도록 한다. 그림 6은 서브 워크플로우와 다중 워크플로우의 추상적인 프로세스를 보인다.

그림 6과 같이 다수의 사용자는 각기 자신의 스케줄에 따라 자신이 필요로 하는 로봇 서비스를 로봇 응용 개발자가 작성한 서브 워크플로우를 이용하여 기술할 수 있다. 예를 들어 그림 6에서 각 사용자의 스케줄에 따른 서비스는 Kim, Lee, Park 각각의 플로우로 표현할 수 있다. 각 사용자의 스케줄 중에 로봇 서비스가 필요한 경우는 서브 워크플로우(Robot Service Flow A, B)를 이용하여 실제 로봇 서비스를 호출할 수 있다.

CAWL에서 제공하는 서브 및 다중 워크플로우 기능은 일정이 서로 다른 사용자에게 각기 다른 워크플로우 서비스를 지원한다. 그리고 로봇 응용 개발자가 기존에 작성한 워크플로우를 재사용함으로서 로봇 서비스 제공을 위한 워크플로우를 용이하게 작성할 수 있는 장점을 가진다. 그림 7은 이와 같은 기능을 지원하는 상황인지기반 워크플로우 시스템의 구조를 나타낸다.

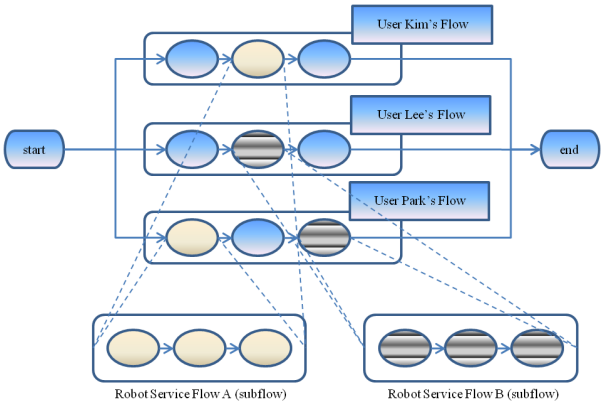


그림 6. URC 로봇 서비스의 추상적 프로세스 정의

그림 7과 같이 상황인지 워크플로우 시스템은 컨텍스트 엔진과 워크플로우 실행 엔진의 2가지로 구분할 수 있다. 켄텍스트 엔진(Context Engine)은 컨텍스트 정보(user profile, device state, location 등)를 RDF 형식을 이용하여 {주어, 동사, 목적어} 단위의 집합으로 표현하는 기능을 수행한다. 그림 8은 RDF 형식으로 기술된 간단한 컨텍스트 모델을 보인다[27].

그림 8은 RDF 기반 컨텍스트 모델의 일부로서, 주어, 동사, 목적어 단위의 집합으로 표현하여 컨텍스트를 기술한다. 그림 8은 “Persion에 해당하는 사람이 OfficeRoomNum에 위치하는가?”를 판단하는 내용을 기술한다. 컨텍스트 엔진은 사용자나 로봇의 주변에 발생하는 낮은 수준의 센싱 정보(Context)를 높은 수준의 상황 정보(Situational Info)로 변환한다. 그리고 CAWL 문서는 기술되어 있는 사용자의 스케줄과 높은 수준의 상황 정보, OWL (Web Ontology Language)[28]로 기술된 상황 모델(Context Model)을 참조하여 사용자에게 필요한 로봇 서비스를 제공한다. 워크플로우 실행 엔진(Workflow Execution Engine)은 사용자의 일정이 작성된 CAWL 로봇 응용 문서와 상황 정보, 컨텍스트 모델(Context Model)에 따라 로봇 서비스를 제공한다.

```
<rule contribute="">
  <constraint name="c1">
    <subject type="Persion"> ?=inVar/persion </subject>
    <verb>locatedIn</verb>
    <object type="Room">?officeRoomNum</object>
  </constraint>
 </rule>
```

그림 8. RDF 기반 컨텍스트 모델

4. 시나리오 적용 예

본 장에서는 유비쿼터스 컴퓨팅 환경에서로봇 서비스의 개발 과정을 제안하는 SOWL에서의 각각의 개발자 계층과 연계하여 설명하도록 한다.

SOWL은 디바이스 소프트웨어, 로봇 소프트웨어, 로봇 응용 소프트웨어, 시나리오 개발자 계층에 독립적인 실행 환경과 개발 환경을 제공하여 모듈성을 높이는 특징을 갖는다. 디바이스 소프트웨어 개발자는 로봇에 디바이스를 제어하거나 센싱 정보를 받아오는 소프트웨어를 개발한다. 잔디 로봇과 문을 열어주는 로봇에서 사용될 수 있는 디바이스는 로봇 팔이나 모바일 구동부, RFID 센서, GPS 센서, 자동 분무기 등이 있다. 이와 같은 디바이스는 이질적인 H/W 플랫폼을 기반으로 서로 다른 업체에서 개발된다. 이 때 개발 업체는 각 디바이스를 제어하기 위한 XML 스키마를 개발하여 소프트웨어와 함께 제공한다. 로봇 소프트웨어 개발자가 SOWL을 이용하였을 때 디바이스 소프트웨어에 대한 소스 코드를 학습할 필요가 없고, XML의 조합으로 구성된 로봇 디바이스 서비스를 설계할 수 있다.

본 단락은 그림 2의 내용을 통해 로봇 시나리오 문서가 작성되어 실제 로봇 디바이스가 제어되는 과정에 대하여 설명한다. 로봇 서비스 시나리오는 유비쿼터스 컴퓨팅 환경과 상황 정보에 대한 전문 지식이 있는 개발자가 작성하는 XML 기반의 문서이다. 시나리오 문서는 상황인지 워크플로우 언어인 CAWL을 이용하여 작성되는 것이며, 사람의 사고과정과 가장 유사하면서 간단히 표현할 수 있는 일차 술어 논리(first order logic)을 기반으로 상황정보가 표현된다[13]. 이와 같이 작성된 시나리오 문서의 내용은 사람의 사고 수준에서 이해할 수 있는 추상적인 수준의 내용으로 작성되며, 로봇 응용 계층의 개발자가 개발한 상황인지 워크플로우 시스템[27]을 통해 처리된 결과는 로봇 소프트웨어 개발 계층으로 전달된다. 이때 처리된 결과의 내용은 실제 로봇 디바이스를 제어하기 위한 제어 정보를 담고 있는 XML 형태의 메시지가 된다. 이와 같이 전달된 로봇 제어 정보는 로봇 각각의 디바이스를 제어하기 위한 소프트웨어를 통해 로봇의 팔, 모바일 구동부와 같은 각각의 디바이스를 제어하게 된다. 이때 로봇 디바이스를 제어하는 로봇 클라이언트 서비스와 로봇 제어를 위한 명령어를 전달하는 로봇 소트트웨어 계층의 로봇 서비스는 R-OSGi를 기반으로 상호 연동한다. 이 때 최종적으로 전달되는 로봇 제어 정보는 실제 로봇 디바이스를 제어하기 위한 구체적인 로봇 제어 명령어를 포함한다.

그림 9는 디바이스 서비스가 각 디바이스에게 디바이스 소프트웨어에 해당 제어 데이터 XML을 전달하거나 받는 구조이다. 그림 9와 같이 디바이스 서비스는 상위 계층인 로봇 서비스가 생성한 로봇 제어 XML을 테그에 따라 분할하여 각 디바이스에게 전달한다. 이와 같은 구조는 각 디바이스 소프트웨어를 제어하기 위해 소스 코드 수준에서 데이터를 분할하는 방법보다 낮은 결합도를 보장한다.

로봇 응용 서비스는 SOMAR 로봇 응용 서버에서 실행되는 로봇 서비스와 로봇 응용 소프트웨어를 호출한다. 로봇 서비스는 디바이스 서비스를 호출하며 SOMAR 로봇 클라이언트의 디바이스 서비스와 로봇 응용 서비스간의 원격 서비스 호출을 담당한다. URC 로봇은 로봇에서 동작해야 하는 소프트웨어 중 일부를 외부 디바이스로 분

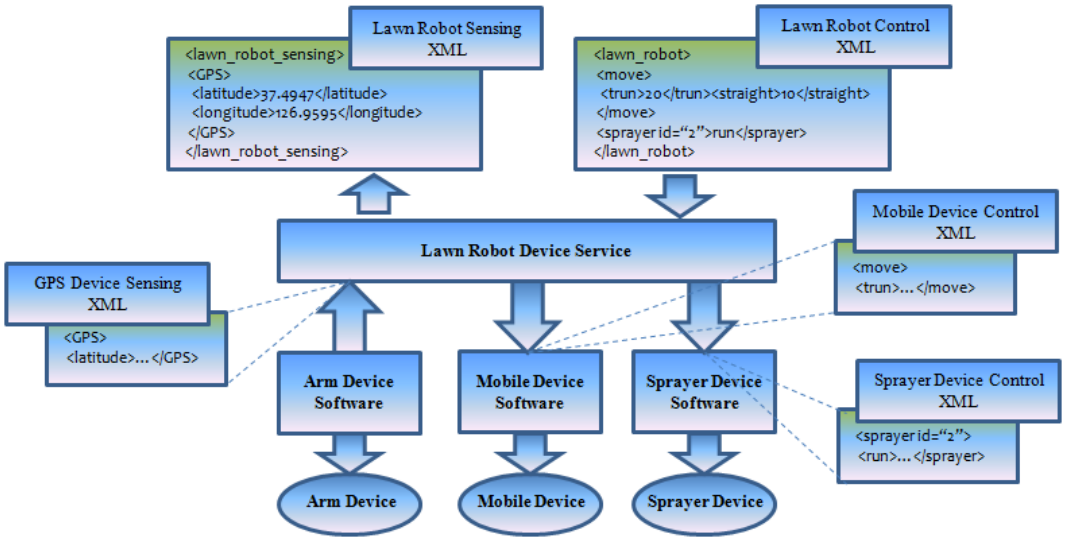


그림 9. 디바이스 서비스와 디바이스 소프트웨어의 관계

담시켜 로봇 하드웨어나 소프트웨어를 단순화하고, 로봇의 서비스가 외부 소프트웨어에 다른 서비스를 요청할 수 있는 다양한 환경을 구축할 수 있다. 로봇 응용 소프트웨어는 로봇에서 동작할 소프트웨어 중 일부로서 로봇 응용 서버에서 동작한다. 그림 10은 잔디 로봇의 로봇 응용 서비스와 로봇 응용 소프트웨어, 로봇 서비스 그리고 디바이스 서비스의 구조를 보인다.

잔디 로봇의 로봇 응용 소프트웨어는 분무를 위한 농약의 농도를 조절하는 소프트웨어 등이 있을 수 있다. 즉, 로봇 응용 소프트웨어는 로봇의 요구사항 중 로봇 디바이스를 직접 제어하지 않는 기타의 소프트웨어가 된다.

상황인지 정보는 추상화된 사용자의 주변환경 정보이지만, 최종 사용자가 이해하기는 어려워 이에 대한 이해도가 높은 시나리오 개발자가 작성해야 한다. 상황인지 기반의 시나리오는 예를 들어 잔디를 관리하는 로봇과 장애인 사용자를 위한 문 열기 로봇을 이용한 환경이 있다.

문을 열어주는 로봇은 사용자가 가지고 있는 RFID가 특정 문의 위치와 가까워지는 경우 사용자를 위해 문을 열어준다. 사용자가 현관에 접근할 경우 문을 열어주고 정원으로 사용자가 이동할 것을 알려준다. 이와 같이 사용자의 상황 정보가 복잡하고 서비스 간에 연관 관계가 있는 경우 최종 사용자가 워크플로우를 작성하기 어렵다. 그러므로 서브 워크플로우를 이용한 시나리오 개발은 최종 사용자에게 URC 로봇을 쉽게 사용할 수 있도록 한다. 그림 11은 로봇 시나리오 개발자가 작성한 서브 워크플로우와 워크플로우 시스템을 보인다.

그림 11과 같이 시나리오 개발자는 상황 정보에 따른

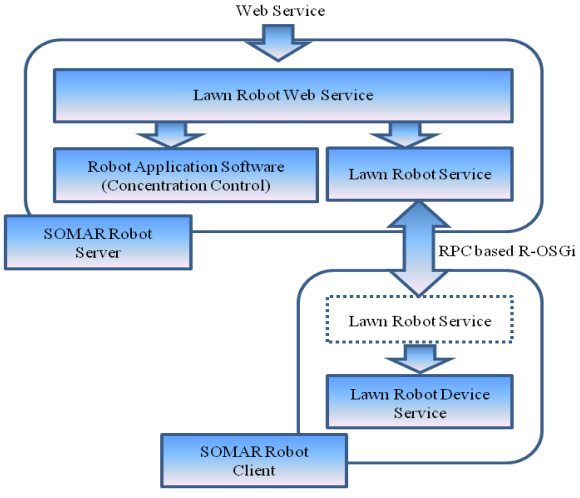


그림 10. 잔디 로봇의 로봇 응용 서비스

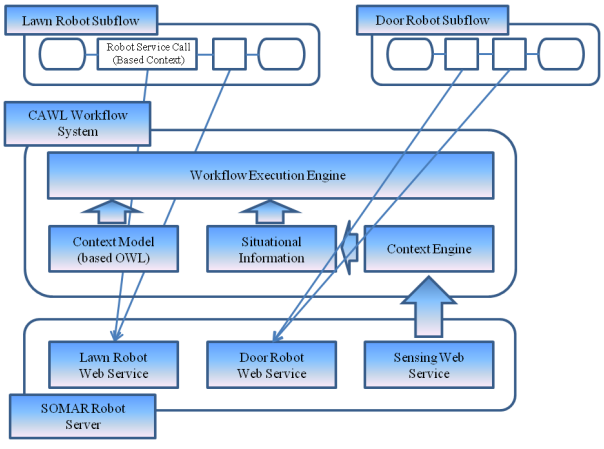


그림 11. 시나리오 개발자가 작성한 서브 워크플로우와 워크플로우

로봇 서비스의 호출을 서브 워크플로우로 작성한다. 또한 센싱 서비스가 생성한 센싱 데이터는 컨텍스트 엔진에 의해 분석되고, 워크플로우 실행 엔진은 컨텍스트 모델과 상황 정보에 따라 로봇 서비스를 실행시킨다. 이와 같은 로봇 시나리오는 최종 사용자에 의해 자신의 스케줄에 따라 재사용된다.

그림 12는 사용자가 작성한 워크플로우와 시나리오 개발자가 작성한 서브 워크플로우의 관계를 보여준다.

최종 사용자는 시나리오 개발자가 작성한 워크플로우를 이용하여 손쉽게 자신의 스케줄에 맞는 워크플로우를 작성할 수 있다.

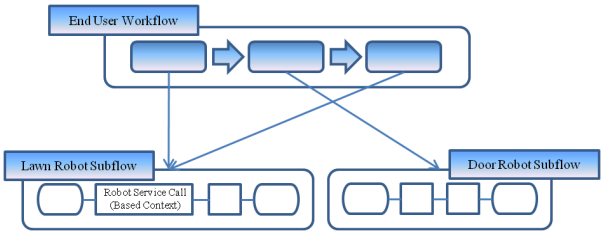


그림 12. 최종 사용자 워크플로우와 서브 워크플로우의 관계

5. 결 론

본 논문에서는 지능형 서비스 로봇의 소프트웨어를 다중 계층화된 SW 플랫폼 구조를 기반으로 하고, 각각의 계층에서의 개발자가 갖는 특성을 고려한 URC 로봇 소프트웨어 아키텍처(SOWL)를 제안하였다. SOWL은 서비스 지향적인 로봇 소프트웨어 미들웨어인 SOMAR 와 로봇 응용 서비스를 기술하기 위한 상황인지 워크플로우 언어인 CAWL을 주요 구성 요소로 갖는다.

제안하는 아키텍처를 기반으로 각각의 개발 계층에서 개발자가 작성한 소프트웨어는 S/W의 모듈화를 높일 수 있고, CAWL을 통해 상황정보에 따라 로봇 서비스를 사용자에게 제공할 수 있다. 이와 같은 기반 기술을 바탕으로 제안하는 SOWL은 사용자 및 응용 계층에서 물리적인 디바이스 플랫폼의 이질성에 대한 고려사항을 최소화하여 약결합(loosely-coupled) 형태의 개발을 지원할 수 있는 장점을 갖는다. 다시 말해서 각각의 소프트웨어 특성에 적합한 개발 계층을 SOWL을 통해 제공함으로써, 개발자가 다른 계층에서 개발한 소프트웨어나 시나리오에 대한 높은 수준의 학습을 요구하지 않고, 자신의 요구사항을 소프트웨어에 적용할 수 있다.

참고문헌

[1] 김성훈, 김종배, “URC를 위한 로봇 S/W 아키텍처 기술,” 대한전자공학회 특집호 제33권, 제3호, pp. 56-63, 2006.

[2] 정승욱, 이승익, 김성훈, “네트워크 로봇을 위한 로봇 소프트웨어 플랫폼에 대한 연구”, 정보과학회지 제26권, 제4호, pp.38-48, 2008.

[3] 홍성수, “RSCA : 분산 로봇 플랫폼에서 임베디드 소프트웨어의 동적 재구성을 지원하는 통합 미들웨어,” 한국통신학회지(정보와통신), 제21권 10호, pp.22-35, 2004. 10.

[4] Hyun Kim, Young-Jo Cho, Sang-Rok Oh, “CAMUS : A middleware supporting context-aware services for network-based robots,” IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO2005), pp.237-242, 2005.

[5] V. K. Murthy, E. V. Krishnamurthy, “Contextual Information Management Using Contract-Based Workflow,” Proc.ACM Computing Frontiers, CF’05, Iscra, Italy, 2005.

[6] Lorenzo Fluckiger, V. To, H. Utz, “Service Oriented Robotic Architecture Supporting a Lunar Analog Test,” International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS), 2008.

[7] Mattias Lindstrom, A. Oreback, H. Christensen, “BERRA : A Research Architecture for Service Robots,” In International Conference on Robotics and Automation, 2008.

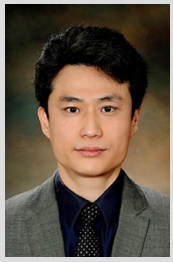
[8] Yinong Chen, W. T. Tsai, “Development of a Security Robot in Service-Oriented Architecture,” http://asusrl.eas.asu.edu/srlab/Research/RoboticsChallenge.html.

[9] Simeon Tsvetanov, “Using Some Motion Devices for Easily Workflows IIlustration,” International Scientific Conference Computing Science’2008, 2008.

[10] Bilge Mutlu, Jodi Forlizzi, “Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction,” Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction, pp. 12-15, March 2008.

[11] 김수연, 황석찬, 곽동규, 최재영, “URC 로봇을 위한 서비스 지향적 서버-클라이언트 미들웨어 아키텍처

설계," 한국 정보과학회 HPC연구회 동계 학술발표대회, pp.21-26, 2009. 2.
[12] 손은미, 곽동규, 황석찬, 최재영, "URC 로봇 클라이언트를 위한 서비스 지향적 디바이스 아키텍처 설계," 한국 정보과학회 HPC연구회 동계 학술발표대회, pp.121-128, 2009. 2.
[13] 최종선, 조용윤, 최재영, "다중-워크플로우를 지원하는 상황인지 워크플로우 언어의 설계," 한국인터넷정보학회 논문지 제10권 제6호, pp.145-157, 2009. 12.
[14] 김현, 조영조, 오상록, "URC (Ubiquitous Robotic Companion): 네트워크 기반 서비스 로봇," 한국정보과학회, 제24권, 제3호, pp.5-11, 2006.
[15] J. Lee, J.-Y. Park, S. Han, and S. Hong. "RSCA: Middleware Supporting Dynamic Reconfiguration of Embedded Software on the Distributed URC Robot Platform," The First International Conference on Ubiquitous Robots and Ambient Intelligence (ICURAI), pp. 426--437, December 2004.
[16] JESS (Java Expert System Shell), http://www. jessrules.com.
[17] VPL (Visual Programming Language), http:// msdn.microsoft.com/en-us/library/bb964572.aspx
[18] BPEL, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
[19] Rosenberg,F., Dustdar. S, "Business rule integration in bpel – a service-oriented approach," In Proc. of the 7th Int. IEEE Conf. on E- Commerce Technology, 2005.
[20] J. Shen, Y. Yang, "From BPEL4WS to OWL-S: Integrating E-Business Process Descriptions," In SCC '05: Proceedings of th 2005 IEEE International Conference on Services Computing, pp.181-190, 2005.
[21] OPRoS, http://opros.or.kr.
[22] MSRDS, http://msdn.microsoft.com/en-us/robotics.
[23] ERSP, http://www.evolution.com/products/ersp.
[24] Satish, Thatte, "XLANG : Web Services for Business Process Design," Microsoft Corp., 2001.
[25] Matthias W., Oliver K., "Towards Context-aware Workflows," In: Pernici, B., Gulla, J.A. (eds.) CAiSE 2007 Proc. of the Workshops and Doctoral Consortium, Vol.2. Tapir Acasemic Press, 2007.
[26] JWS (Java Web Services), http://java.sun.com/webservices/docs/1.6/tutorial/doc.
[27] 최종선, 조용윤, 최재영, "복합 워크플로우 서비스를 위한 CAWL 기반 상황인지 워크플로우 시스템," 한국정보처리학회, 제17-A권 제2호, pp.93-102, 2010. 4.
[28] OWL (Web Ontology Language), http://www.w3.org/TR/ owl-features/.



곽 동 규

2001 서경대학교 응용수학과(학사)
2004 숭실대학교 컴퓨터학과(공학석사)
2004~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 프로그래밍 언어, 컴파일러, XML

최 종 선

2000 숭실대학교 컴퓨터학부(학사)
2002 숭실대학교 컴퓨터학과(공학석사)
2003~현재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 분산 컴퓨팅, 유비쿼터스 컴퓨팅, 지능형 로봇 미들웨어

최 재 영

1984 서울대학교 제어계측공학과(학사)
1986 미국 남가주대학교 컴퓨터공학(석사)
1991 미국 코넬대학교 컴퓨터공학(박사)
1992~1994 미국 국립 오크리지연구소 연구원
1994~1995 미국 테네시 주립대학교 연구교수
2001~2002 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원
1995~현재 숭실대학교 정보과학대학 컴퓨터학부 교수
관심분야 : 분산 컴퓨팅, 고성능컴퓨팅(HPC), 유비쿼터스 컴퓨팅

유 재 우

1976 숭실대학교 전자계산학과(학사)
1978 한국과학기술원 전산학과(석사)
1985 한국과학기술원 전산학과(박사)
1983~현재 숭실대학교 정보과학대학 컴퓨터학부 교수
관심분야 : 프로그래밍 언어, 컴파일러, 인간과 컴퓨터 상호작용