http://dx.doi.org/10.3745/KIPSTA.2012.19A.1.001

지능형 네트워크 로봇을 위한 서비스 지향적인 로봇 클라이언트 미들웨어 설계와 구현

곽 동 규[†] · 최 재 영[††]

요 약

지능형 네트워크 로봇은 다양한 환경에서 네트워크 시스템과 연계하여 인간과 상호작용을 하며, 상황에 따라 주어진 역할을 수행한다. 유비쿼터스 환경에서 동작하는 네트워크 기반의 URC 로봇은 분산 컴퓨팅 환경에서 클라이언트 로봇의 기능을 서버로 분산시킴으로써 클라이언트 로봇을 경량화하는 장점을 갖는다. URC 로봇 환경 중에서 SOMAR는 서버-클라이언트 환경에서 서비스 지향기법으로 로봇 소프트웨어를 개발하기 위해 제안되었다. 본 논문에서는 URC 로봇 환경에서 사용 가능한 SOMAR 로봇 클라이언트를 소개하고 그 구현을 보인다. SOMAR 로봇 클라이언트는 디바이스 서비스 계층과 로봇 서비스 계층을 갖는다. 이 중 디바이스 서비스는 디바이스를 제어하는 서비스이고, 로봇 서비스는 다수의 디바이스 서비스를 결합하여 생성된 로봇이 제공하는 서비스를 추상화시킨 것이다. 또한 본 논문에서는 디바이스와 로봇 서비스의 결합 관계를 표현하기 위해 RSEL (Robot Service Executing Language)을 이용하였다. 서비스 결합을 기술한 RSEL 문서는 변환기를 통해 클라이언트 시스템 언어로 변환하고 컴파일링하여 로봇 클라이언트 시스템에 업로드한다. SOMAR 클라이언트 시스템은 호스트/타겟 구조를 갖는 내장형 시스템에 적용하기가 용이하며, RSEL 처리 엔진에 대한 부담을 줄여서 로봇 클라이언트를 경량화시켰다.

키워드 : 유비쿼터스, 지능형 네트워크 로봇, URC, 서비스 지향, 로봇 클라이언트, SOMAR

A Design and Implementation of A Robot Client Middleware for Network-based Intelligent Robot based on Service-Oriented

Donggyu Kwak[†] · Jaeyoung Choi[††]

ABSTRACT

Network-based intelligent robot is connected with network system, provides interactions with humans, and carries out its own roles on ubiquitous computing environments. URC (Ubiquitous Robot Companion) robot has been proposed to develop network-based robot by applying distributed computing techniques. On URC robot, it is possible to save the computing power of robot client by environments, has been proposed to develop robot software using service-oriented architecture on server-client computing environments. The SOMAR client robot consists of two layers - device service layer and robot service layer. The device service controls physical devices, and the robot service abstracts robot's services, which are newly defined and generated by combining many device services. RSEL (Robot Service Executing Language) is defined in this paper to represent relations and connections between device services and robot services. A RESL document, including robot services by combining several device services, is translated to a programming language for robot client system using RSEL translator, then the translated source program is compiled and uploaded to robot client system with RPC (Remote Procedure Call) command. A SOMAR client system is easy to be applied to embedded systems of host/target architecture. Moreover it is possible to produce a light-weight URC client robot by reducing workload of RSEL processing engine.

Keywords : Ubiquitous, Network-based Intelligent Robot, URC, Service-oriented, Robot Client, SOMAR

1. 서 론

최근 눈부시게 발전하고 있는 IT 기술을 바탕으로 가사, 오락, 공공 서비스와 같은 다양한 서비스를 제공할 수 있는 네트워크 로봇 (ubiquitous-로봇, u-로봇)에 대한 관심이 고조되고 있다[1]. 이를 위하여 분산 컴퓨팅 환경을 네트워크 로봇에 적용하여 서버-클라이언트 구조의 URC (Ubiquitous

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음(NIPA-2011-C1090-1121-0010).
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 정보과학대학 컴퓨터학부 교수
논문접수 : 2011년 8월 19일
수 정 일 : 1차 2011년 11월 14일
심사완료 : 2011년 11월 14일

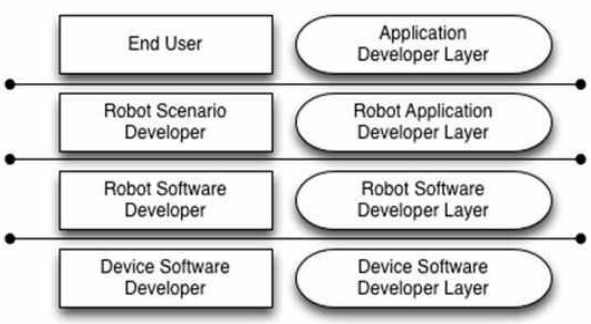
Robotic Companion) 로봇[1, 2, 3]이 제안되었다. URC 로봇은 “언제 어디서나 우리 주변에서 서비스를 지원하는 로봇”을 의미한다. 분산 컴퓨팅 환경에서 URC 클라이언트 로봇의 소프트웨어를 서버로 분산시킬 수 있으므로, 낮은 사양의 하드웨어를 이용하여 클라이언트 로봇을 구현할 수 있다. 이는 클라이언트 로봇의 생산원가를 절감시키는 효과로 이어진다. 또한 단일 제어 서버를 통해 다수의 로봇에게 서비스를 제공할 수 있다.

현재 URC 로봇에 관한 대표적인 연구는 서울대학교에서 연구한 RSCA (Robot Software Communications Architecture)[4, 5]와 한국전자통신연구원에서 연구한 uROSE (urc RObot Service development Environment)[2]가 있다. 이 두 연구는 로봇에서 사용하는 소프트웨어를 추상화된 계층으로 구분하여 제안하였다. 이는 개발자로 하여금 각 요소 개발에 집중할 수 있는 장점을 가지지만 각 응용간의 관계를 응용 프로그램 코드로 직접 작성해야 한다. 이는 응용 프로그램의 복잡도를 증가시키는 요인이 될 수 있다.

서비스 지향 구조 (SOA : Service Oriented Architecture)는 소프트웨어 개발 방법론 중에서 다양한 환경에서 응집도를 높이고 결합도를 낮추는 방법론이다. 서비스 지향 구조는 프로그램의 재사용성을 높이고 각기 다른 환경의 응용간의 인터페이스를 제공한다. 로봇 환경은 로봇의 특징이나 제조사에 따라 다양한 플랫폼이 존재한다. 또한 로봇은 각 로봇마다 다른 디바이스를 포함하고 있으며, 그 디바이스로 서비스를 제공한다. 그러므로 로봇 환경에서 개발자는 다양한 플랫폼에서 동작하는 디바이스에 종속적인 프로그램을 작성해야 한다. 이는 프로그램의 복잡도를 높일 뿐만 아니라 프로그램의 재사용성을 떨어뜨린다. 이에 따라 로봇 환경에 적합한 구조에 관한 연구가 활성화되고 있다. 서비스 지향 구조를 적용한 SORA (Service Oriented Robotic Architecture)[6]는 로봇이 요구하는 다수의 응용 프로그램을 서비스로 정의하고 이를 위한 인터페이스를 제안하였다. SORA는 다수의 개발자가 존재하는 로봇 소프트웨어 개발에 효율적이지만, 로봇이 보유하는 디바이스에 관한 추상화가 부족하여 디바이스 서비스의 재사용성이 낮다.

우리는 서비스 지향 구조를 적용하여 URC 로봇을 개발하는 연구를 진행해왔다. SOWL (SOMAR and CAWL)[7]은 URC 로봇 소프트웨어 개발자를 디바이스 소프트웨어 개발와 로봇 소프트웨어 개발자, 시나리오 개발자, 사용자로 구분하고 각 개발자 계층에게 모듈화를 위한 소프트웨어 실행 계층을 제공한다. 디바이스 개발자는 디바이스를 제어하는 소프트웨어를 개발하고, 로봇 소프트웨어 개발자는 디바이스 소프트웨어를 조합하여 로봇 소프트웨어를 개발한다. 그리고 시나리오 개발자는 URC 로봇을 유비쿼터스 환경에서 상황 정보에 따라 로봇 소프트웨어를 호출하는 워크플로우를 작성한다. 사용자는 시나리오 개발자가 작성한 워크플로우를 조합하여 자신의 스케줄에 맞는 워크플로우를 작성할 수 있다. (그림 1)은 로봇 개발환경의 개발자 계층을 보여준다.

SOWL을 이용한 로봇 개발 환경은 SOMAR (Service-Oriented Middleware Architecture for Robot)[8, 9]와 CAWL (Context-Aware Workflow Language)[10] 엔진으로 구성되어 있다. SOMAR는 디바이스 소프트웨어와 로봇 소프트웨어의 실행 계층이며, CAWL은 상황인지 기반의 워크플로우 언어이다. SOMAR는 디바이스 서비스와 로봇 서비스들간의 호출 관계를 정의하는 RSEL (Robot Service Executing Language)을 이용하여 로봇의 요구사항을 웹 서비스로 제공한다. RSEL은 디바이스 서비스와 로봇 서비스 간의 결합관계를 표현하는 언어이다. 그리고 CAWL을 이용하여 서비스의 플로우를 기술한다. 이와 같은 환경은 각 개발자 계층에 독립적인 개발 환경을 제공하여 개발자가 다른 계층에서 개발한 소프트웨어나 시나리오에 대한 높은 수준의 학습을 요구하지 않고, 자신의 요구사항을 소프트웨어에 적용시킬 수 있는 장점을 가진다.



(그림 1) 로봇 개발환경의 개발자 계층

서비스를 이용한 개발 방법은 높은 응집도와 낮은 결합도를 보장하여 소프트웨어의 재사용성을 높이고, 프로그램간의 간섭에 의한 부작용을 줄여 소프트웨어의 신뢰성을 높인다. 본 논문에서는 서비스 지향 구조를 이용한 SOMAR 로봇 클라이언트를 제안한다. SOMAR 로봇 클라이언트는 로봇이 설치되어 있는 디바이스와 응용 프로그램을 서비스로 정의한다. 또한 제공하는 서비스간의 결합을 추상적으로 표현하기 위해 RSEL 문서를 사용하였다. RSEL은 디바이스 프로그램과 로봇 응용 프로그램의 독립성을 보장하고 다수의 디바이스 서비스를 묶어 로봇 서비스를 생성할 수 있는 추상적인 인터페이스를 제공하여 효과적으로 로봇 클라이언트를 개발할 수 있는 장점을 가진다. 그리고 디바이스 서비스와 로봇 응용 서비스로 구성되어 있는 로봇 서비스를 URC 서버에서 RPC (Remote Procedure Call)로 호출 가능하도록 설계하여 URC 환경에서 사용하기가 용이하다. SOMAR 로봇 클라이언트에서 RSEL은 로봇 클라이언트에서 동작하는 언어로 변환하여 적용한다. 이는 로봇 클라이언트에서 RSEL을 처리하기 위한 엔진을 배제하여 로봇 클라이언트를 경량화시킬 수 있다.

본 논문은 2장에서 관련연구를 보인 후 3장에서 SOMAR 로봇 클라이언트의 개념을 소개한다. 그리고 4장에서 SOMAR 로봇 클라이언트에서 서비스의 결합을 정의하는 RSEL의 설계에 관해 논한 후 5장에서 시스템 구조를 보인다. 마지막으로 6장에서 결론짓는다.

2. 관련 연구

본 연구의 목적은 다수의 디바이스가 탑재된 URC 로봇 클라이언트 개발 환경에서 높은 모듈화를 제공하고 신뢰도를 향상시키기 위해서 서비스 지향적인 소프트웨어 구조를 URC 로봇 클라이언트 개발 환경에 적용하는데 있다. 본 장에서는 기존 URC 로봇 환경에 관한 내용과 로봇이나 디바이스 소프트웨어를 위한 서비스 지향적인 구조에 대한 연구를 소개한다.

2.1 URC 로봇과 OPRoS 기반의 환경

URC는 언제 어디서나 나와 함께 하고, 나에게 필요한 서비스를 제공하는 것을 목적으로 하며, 유비쿼터스 환경에서 동작하는 네트워크 기반의 지능형 로봇이다. 현재 URC 로봇에 관한 연구는 정부를 중심으로 학계, 연구소, 산업계가 공동으로 OPRoS (Open Platform for Robotic Service)[11]를 제안하였으며 이를 적용한 uROSE나 RSCA연구가 진행중이다. OPRoS는 URC에서 요구하는 대부분의 개념을 표준화하여 제공하고 있지만, 구체적인 통신 프로토콜이나 소프트웨어 구조를 미들웨어 수준에서 규정하고 있지는 않는다.

RSCA는 URC의 분산 로봇 플랫폼에서 내장형 소프트웨어의 동적 재구성을 지원하는 통합 미들웨어이다. RSCA에서 URC 로봇의 핵심적인 특징은 URC 서비스 사업자들이 제공하는 고용량 서버를 로봇의 움직임을 제어하기 위하여 활용하고, 홈네트워크에 연결된 다양한 정보 가전기기 및 센서 네트워크등을 로봇이 활용한다는 점에 두었다. (그림 2)는 RSCA가 제안한 URC 로봇의 하드웨어와 소프트웨어 구성을 보인다.

RSCA는 코어 프레임워크에 해당하는 응용 소프트웨어 계층과 실시간 운영체제, 분산 미들웨어의 구조를 제안하였다. 그 중 응용 소프트웨어는 계획 계층 (Decision Planning Layer)와 테스크 계층 (Task Execution Layer), 행위 수행 계층 (Behavioral Execution Layer), 하드웨어 추상화 계층 (Hardware Abstraction Layer)으로 구성하여 각 계층별로 역할을 분담하였다. RSCA를 이용한 로봇 소프트웨어 개발자는 하드웨어 장치 개발자와 응용 컴포넌트 개발자, 응용 개발자로 분류할 수 있다. 응용 소프트웨어 계층은 개발자에게 해당 계층을 제공하여 각 요소에 집중할 수 있는 장점을 가지지만, 응용간의 관계를 응용 프로그램 코드에 직접 기술하여야 한다. 이는 응용 프로그램의 복잡도를 증가시키는 약점을 가진다.

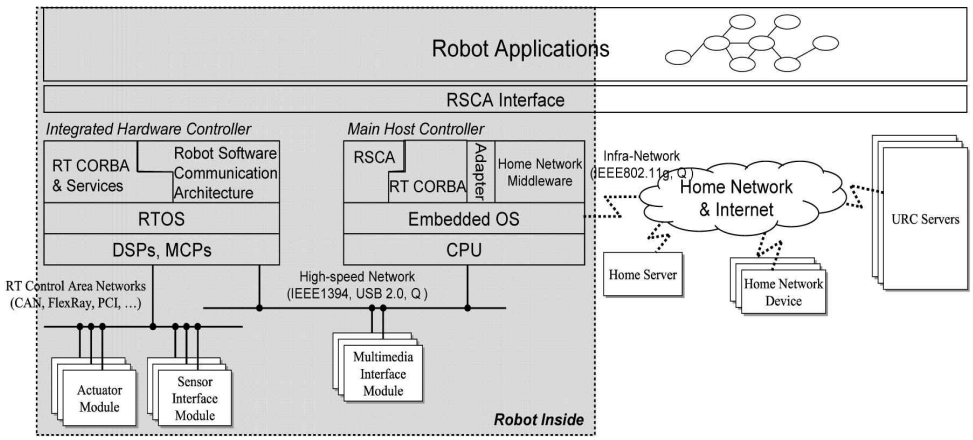
2.2 SORA

미국항공우주국 (NASA : National Aeronautics and Space Administration)에서는 다수의 개발자가 다른 팀에 소속되어 서로 다른 로봇 개발 환경에서 프로그램을 개발해야 하는 어려움을 극복하기 위해 SORA를 연구하였다. 그리고 SORA를 HRSS (Human-Robot Site Survey and Sampling for Space Exploration)[12] 프로젝트에서 테스팅하였다. HRSS는 무인 로봇이 달을 탐사하기 위한 프로젝트이다. 이 프로젝트에서는 로봇 환경에 SOA를 적용하여 다수의 응용 프로그램을 개발하였다. (그림 3)은 HRSS의 한 모델인 K10에 적용한 서비스를 보여준다.

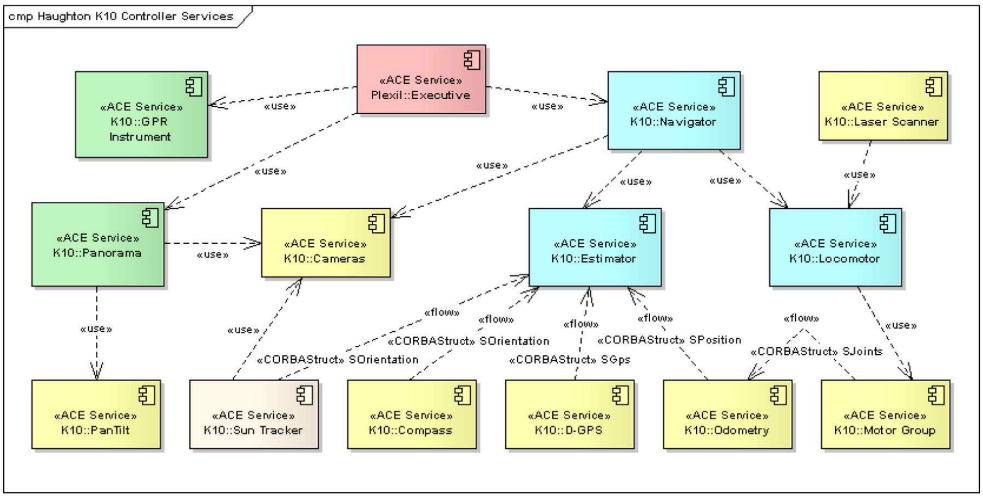
(그림 3)과 같이 로봇의 기능을 서비스 단위로 개발하고, 각 서비스와 다른 서비스간의 관계를 분석하여 사용(use)과 흐름(flow)으로 전체 로봇의 응용 소프트웨어를 개발하였다. 이와 같은 개발 방법은 서비스간의 결합도를 낮추어 다수의 개발자가 존재하는 로봇 소프트웨어 개발에 효율적이다. 하지만 로봇이 보유하고 있는 디바이스에 대한 추상화가 부족하고 서비스간의 관계를 독립적으로 기술할 수 있는 방법을 제공하고 있지 않아 응용 프로그램의 복잡도를 증가시키는 약점을 갖는다.

2.3 SODA (Service Oriented Device Architecture)

SODA는 물리적인 디바이스를 추상화하여 SOA를 적용한 디바이스 응용 프로그램의 개발 방법이다[13]. 일반적으로 디바이스 응용 프로그램 특성은 디바이스가 가지고 있는 물리적인 특성에 따른다. 즉, 디바이스 응용 프로그램은 디바이스



(그림 2) URC 로봇의 하드웨어와 소프트웨어 구성



(그림 3) SORA를 적용한 로봇의 서비스 예제

가 가지고 있는 특성에 따라 프로그램을 작성해야 한다. 일반적으로 이와 같은 프로그램은 디바이스를 개발한 회사에서 디바이스 드라이버 (Device Driver)와 같은 형태로 배포한다. 하지만 이와 같은 개발 방법은 분산 환경에서는 적합하지 않다. 디바이스 드라이버를 분산 환경에서 사용하기 위해서는, 디바이스를 제어하는 엔터프라이즈 서비스를 생성하고 그 엔터프라이즈 서비스를 이용해야 한다. SODA는 디바이스를 엔터프라이즈 서비스와 동일한 형태로 사용하기 위해 디바이스를 서비스로 정의하였다. 이는 디바이스를 응용 프로그램과 독립적으로 사용할 수 있으므로 재사용성이 우수하다.

SODA는 디바이스 서비스의 표준화된 인터페이스를 위해 두 개의 어댑터를 제안하였다. 그 중 하나는 디바이스의 표준 인터페이스를 위한 디바이스 아댑터 (Device Adapter)이고 다른 하나는 엔터프라이즈 버스 (Enterprise Bus)와의 표준화를 위한 버스 어댑터 (Bus Adapter)이다. 또한 이 두 어댑터를 계층으로 분리하여 서비스에 대한 정보를 디바이스 서비스 레지스트리 (Device Service Registry)에 저장하도록 정의하였다. 두 개의 계층으로 분리된 디바이스 서비스 레지스트리는 다양한 서비스와 서비스 버스에 능동적으로 대응할 수 있는 장점을 갖는다. 하지만 SODA는 디바이스 드라이버를 엔터프라이즈 서비스로 호출할 수 있도록 하여 경량화가 필요한 로봇 클라이언트 환경에는 적합하지 않다.

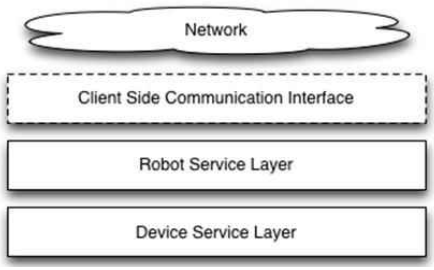
3. SOMAR 로봇 클라이언트

SOMAR는 서비스 지향적인 URC 로봇 개발 환경을 지원하기 위한 미들웨어이다[7, 8, 9]. URC 로봇은 로봇 클라이언트를 하드웨어/소프트웨어적으로 경량화하기 위해 로봇 클라이언트에서 실행해야 할 응용의 일부를 서버에서 실행한다. 또한 유비쿼터스 환경에서는 단일 환경에 다수의 로봇이 서로 정보를 주고 받으며 서비스를 제공할 수 있어야 한다. 그러므로 정보를 통합하고 분석하여 로봇에게 명령을 내려줄 서버가 필요하다. 이에 OPRoS에서는 서버-클라이언트 구조의 URC 로봇을 제안하고 있다.

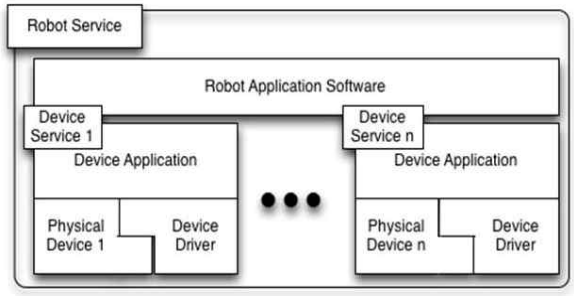
URC 로봇 개발 환경에서의 개발자는 서버 개발자와 클라이언트 개발자로 구분할 수 있고, 이중 클라이언트 개발자는 다시 디바이스 개발자와 로봇 개발자로 분류할 수 있다. 디바이스 개발자는 디바이스 개발사에서 제공하는 디바이스 제어 프로그램을 개발하는 개발자이다. 디바이스 제어 프로그램은 디바이스에 종속적인 프로그램이며, 하드웨어적인 디바이스 제어 신호까지 충분히 이해하고 있어야 개발할 수 있다. 그러므로 디바이스 하드웨어를 개발한 업체에서 디바이스 제어 프로그램도 제공하여야 한다. 또한 로봇 환경은 일반적인 컴퓨팅환경과 달리 다양한 디바이스가 사용되므로 표준 인터페이스를 정의하기가 어렵다. 이와 같은 문제를 해결하기 위해 SOMAR 로봇 클라이언트에서는 서비스 지향적인 프로그래밍 개발 기법을 로봇 디바이스 환경에 적용하여, 디바이스 제어 프로그램을 디바이스 서비스로 정의하였다.

로봇 개발자는 로봇에게 주어진 요구사항을 분석하여 필요로 하는 디바이스를 선택하고, 로봇에서 실행될 응용프로그램을 작성하는 개발자이다. SOMAR 로봇 클라이언트는 다수의 디바이스를 보유하고 있는 로봇의 요구사항에 따라 디바이스 서비스의 결합으로 로봇 서비스를 제공한다. (그림 4)는 SOMAR 로봇 클라이언트가 제공하는 서비스 계층을 보여준다.

SOMAR 로봇 클라이언트의 서비스 계층은 (그림 4)에서 보듯이 디바이스 서비스 계층과 로봇 서비스 계층으로 구성된다. URC 로봇의 로봇 개발자는 로봇에 요구사항에 따라 디바이스를 조합하고 로봇의 응용 프로그램을 작성한다. SOMAR 로봇 클라이언트는 로봇에서 요구되는 소프트웨어를 로봇의 응용 로봇 서비스 (Robot Service)로 정의하고 디바이스 서비스의 조합과 로봇 응용 서비스 (Robot Application Service)로 로봇 서비스를 작성한다. 로봇 응용 서비스는 로봇에서 사용하는 디바이스 제어를 제외한 응용 프로그램의 서비스 단위이다. (그림 5)는 SOMAR 로봇 클라이언트의 로봇 서비스와 디바이스 서비스의 추상적인 구조를 보여준다.



(그림 4) SOMAR 로봇 클라이언트의 서비스 계층



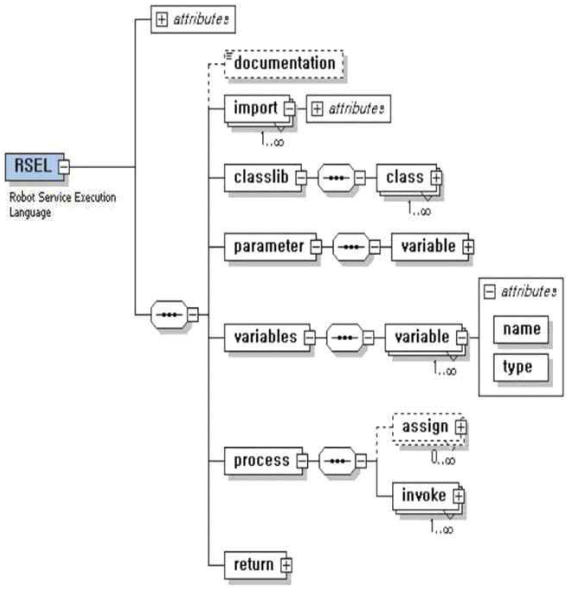
(그림 5) 로봇 서비스와 디바이스 서비스의 관계

(그림 5)와 같이 각각의 디바이스 서비스는 디바이스를 제어하기 위한 디바이스 드라이버와 디바이스 응용 프로그램을 포함한다. 로봇은 다수의 디바이스 서비스를 보유하고 있으며, 이를 결합하여 로봇 서비스로 정의한다. 이와 같은 구조를 이용하면 디바이스 개발자는 디바이스 개발에 그리고 로봇 개발자는 로봇 개발에 집중할 수 있도록 하여 소프트웨어를 쉽게 개발할 수 있다. 또한, SOA 개발 환경은 개발자에게 편리하여 손쉽게 사용이 가능한 장점을 가진다.

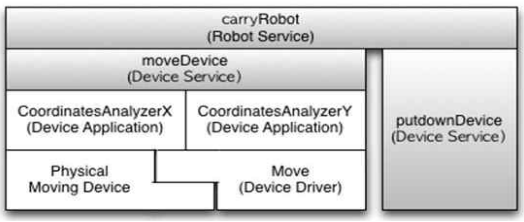
4. RSEL (Robot Service Execution Language)

RSEL은 로봇 서비스간의 결합을 표현하는 언어이다. SOMAR 로봇 클라이언트는 다수의 디바이스 서비스와 로봇 응용 서비스의 결합으로 로봇 서비스를 생성한다. 본 장에서는 디바이스 서비스와 로봇 응용 서비스의 결합을 표현하는 RSEL에 관해 논한다. (그림 6)은 제안하는 RSEL의 스키마이다.

RSEL의 중요 엘리먼트는 다섯 개로 구성되어 있고 각 엘리먼트의 내용은 다음과 같다. “import” 엘리먼트는 서비스의 인터페이스와 타입을 정의하고 “classlib” 엘리먼트는 서비스에 사용되고 있는 라이브러리를 추가한다. 또한 “parameter” 엘리먼트는 해당 서비스의 인자를 나타내고, “variables”는 해당 서비스에서 사용되는 변수이다. 그리고 “process”는 해당 서비스의 동작을 의미하며 “assign”을 통해 변수 값 할당을 다루고 “invoke” 엘리먼트를 통해 다른 서비스를 호출한다. 마지막으로 “return” 엘리먼트는 해당 서비스가 반환하는 값을 의미한다. 본 장에서는 RSEL 문서를 보이기 위해 간단한 서비스 결합 예제를 보인다. (그림 7)은 본 장에서 보이는 예제 시스템의 로봇 서비스 구조를 보인 것이다.



(그림 6) RSEL 스키마



(그림 7) 예제 시스템의 서비스 구조

예제로 보일 로봇 서비스 (carryRobot)는 화물을 나르는 서비스이다. carryRobot은 두 개의 디바이스 서비스의 결합으로 이루어져 있고 디바이스 서비스 중에서 moveDevice는 서비스 공간에 있는 로봇을 다른 서비스가 생성한 절대 위치로 이동시키는 디바이스 서비스이다. 그리고 putdown Device는 화물을 내려놓는 서비스이다. moveDevice는 두 디바이스 응용과 하나의 디바이스를 제어하는 디바이스 드라이버로 구성된다. CoordiatesAnalyzerX와 Coordiates AnalyzerY는 목적 위치 좌표를 이용하여 로봇의 현재 위치에서 목적 위치까지의 상대 위치를 분석하는 응용 프로그램이고 Move는 로봇을 입력되는 위치만큼 이동하는 디바이스 드라이버이다. (그림 8)은 디바이스 서비스를 기술한 RSEL의 예이다.

(그림 8)에서 기술한 moveDevice 디바이스 서비스는 두 개의 디바이스 응용 프로그램과 하나의 디바이스로 구성되어 있다. (그림 7)에서 invoke 엘리먼트는 다른 서비스나 응용 프로그램, 혹은 디바이스를 실행시킨다. 디바이스 서비스에서 응용 프로그램은 디바이스 서비스가 요구하는 프로그램을 의미한다. 또한 디바이스는 실제 장치를 동작시키는 디바이스 드라이버이다. 서비스를 기술한 RSEL 문서는 타겟 시스템에서 추가적인 처리없이 동작하기 위해 타겟 시스템에서 사용하는 프로그래밍 언어로 변환한다. 디바이스 서비스는 결합하여 로봇 서비스를 생성한다. (그림 9)는 로봇 서비스를 기술한 RSEL이다.

```
<RSEL name="moveDevice" layer="deviceService"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:rsel="http://resl.coolman.org/rsel/"
 xmlns:moveDevice="http://device.coolman.org/rsel/ex/moveDevice.wsdl">
 <rsel:import type="http://schemas.xmlsoap.org/wsdl/"
  location="http://device.coolman.org/rsel/ex/moveDevice.wsdl" />
 <variables>
  <variable name="x" type="xsd:int" />
  <variable name="y" type="xsd:int" />
  <variable name="movePoint" type="moveDevice:coordinates" />
 </variables>
 <parameter>
  <variable name="currentCoordinates" type="moveDevice:coordinates" />
 </parameter>
 <process>
  <invoke layer="deviceApp" operation="CoordinatesAnalyzerX"
   inputVariable="coordinates.x" outputVariable="x" />
  <invoke layer="deviceApp" operation="CoordinatesAnalyzerY"
   inputVariable="coordinates.y" outputVariable="y" />
  <assign>
   <from variable="x" /><to variable="movePoint.x" />
  </assign>
  <assign>
   <from variable="y" /><to variable="movePoint.y" />
  </assign>
  <invoke layer="device" operation="move" inputVariable="movePoint" />
 </process>
</RSEL>
```

(그림 8) 디바이스 서비스를 기술한 RSEL

```
<RSEL name="carryRobot" layer="robotService"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:rsel="http://resl.coolman.org/rsel/"
 xmlns:moveDevice="http://device.coolman.org/rsel/ex/moveDevice.wsdl"
 xmlns:putdownDevice="http://device.coolman.org/rsel/ex/secondDevice.wsdl"
 xmlns:carryRobot="http://device.coolman.org/rsel/ex/carryRobot.wsdl">
 <rsel:import type="http://schemas.xmlsoap.org/wsdl/"
  location="http://device.coolman.org/rsel/ex/moveRobot.wsdl" />
 <variables>
  <variable name="movePoint" type="moveDevicecoordinates" />
 </variables>
 <parameter>
  <variable name="servicePoint" type="moveRobot:moveService" />
 </parameter>
 <process>
  <assign>
   <from variable="servicePoint.moveDevice.coordinates" />
   <to variable="movePoint" />
  </assign>
  <invoke layer="deviceService" operation="moveDevice"
   inputVariable="movePoint" />
  <invoke layer="deviceService" operation="putdownDevice"
   inputVariable="servicePoint.putdownDevice.serviceName" />
 </process>
</RSEL>
```

(그림 9) 로봇 서비스를 기술한 RSEL

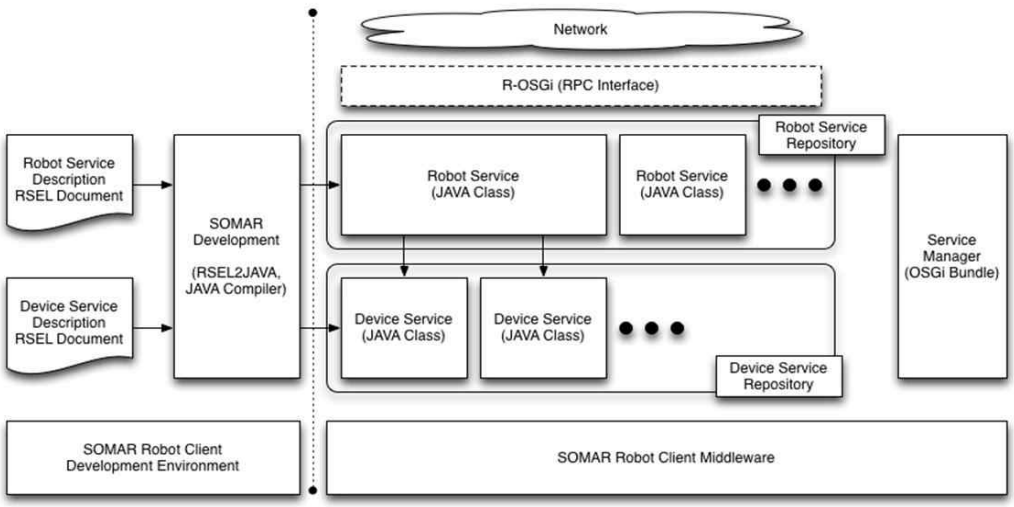
(그림 9)는 디바이스 서비스의 결합으로 표현된 로봇 서비스이며 RSEL로 작성한 로봇 서비스는 디바이스 서비스와 동일하게 타겟 시스템 언어로 변환되고 URC 서버에서 원격 호출 가능하도록 R-OSGi (Remoting-Open Service Gateway Initiative)[14]를 이용하여 서비스로 등록된다. 디바이스 서비스와 로봇 서비스는 동일한 RSEL 문법으로 작성하며 변환기를 통해 변환하여 로봇 클라이언트 시스템에서 제공된다.

5. 시스템 구조

SOMAR 클라이언트 시스템은 서비스 결합을 표현하는 RSEL 문서를 처리하여 클라이언트 시스템에서 사용하는 언어로 변환하고 클라이언트 시스템에 업로드하는 개발환경과 로봇 클라이언트 미들웨어 환경으로 구분되어 있다. SOMAR 로봇 클라이언트 미들웨어 구현 모델은 R-OSGi를 이용한 분산 환경을 제안한다. R-OSGi는 OSGi[15] 기반의 분산 미들웨어 환경이다. (그림 10)은 SOMAR 클라이언트 시스템의 전체 구조를 보인다.

(그림 10)과 같이 디바이스 서비스와 로봇 서비스는 RSEL2JAVA 변환기를 통해 타겟 시스템 언어인 JAVA 프로그램으로 변환되고 컴파일링하여 R-OSGi에 로드하고 등록한다. RSEL2JAVA는 RSEL 문서를 입력으로 JAVA 프로그램을 출력으로 하는 변환기이다. RSEL2JAVA 변환기는 XML 포맷으로 작성된 RSEL 문서를 파싱하여 JAVA 프로그램 소스 코드를 생성한다. (그림 11)은 RSEL2JAVA 변환 알고리즘이다.

(그림 11)의 RSEL2JAVA 알고리즘은 소프트웨어 디자인 패턴 중 Visitor 패턴[16]을 사용하고 있다. 이와 같이 JAVA 프로그램을 생성하는 방법은 교차 개발 환경에 적합하고 로봇 클라이언트에 RSEL 엔진이 필요하지 않아 로봇 클라이언트 시스템을 경량화할 수 있는 장점을 가진다. 생



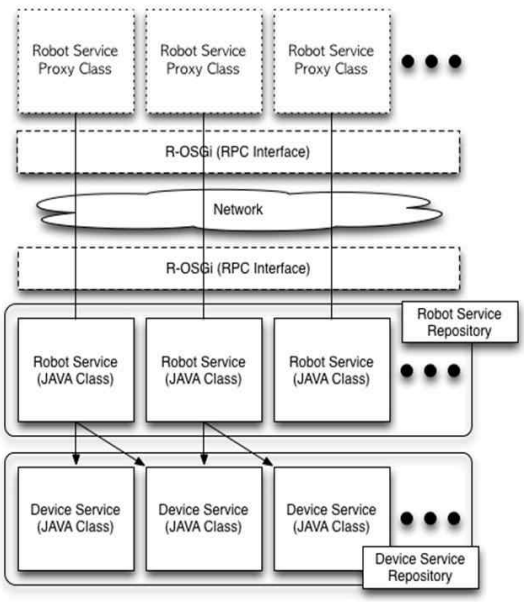
(그림 10) SOMAR 로봇 클라이언트 시스템과 개발환경

```
public class RSEL2JAVA extends Visitor {
  String importString          // 타겟 소스에서 사용하는 패키지
  String variableString        // 타겟 소스에서의 변수 소스
  String bodyString            // 타겟 소스에서 함수 몸체 소스
  Hashtable<String, SOMARType> symbolTable = new Hashtable<String, SOMARType> ();
  public void visit (RSELElement node) {
    // 네임스페이스 분석
    for(int i = 0; i < node.childNodes.size(); i++)
      node.childNodes.elementAt(i).accept(this); }
  public void visit (ImportElement node) {
    importString =+ node.getJAVACode();
    // location 속성의 파일을 분석하여 패키지를 생성하고 경로를 변환 }
  public void visit (VariableElement node) {
    if(부모 엘리먼트가 variables)
      //모듈 안에 변수 생성
    else if(부모 엘리먼트가 parameter)
      // 모듈 인수 생성 }
  public void visit (ProcessElement node) {
    for(int i = 0; i < node.childNodes.size(); i++)
      node.childNodes.elementAt }
  public void visit (AssignElement node) {
    //symbolTable을 참조하여 변수의 타입 케스팅
    //변수 할당을 위한 JAVA 소스 }
  public void visit (InvokeElement node) {
    //layer 속성 값에 해당하는 서비스 저장소에서 서비스를 호출하는 소스 코드 }
  public void visit (ReturnElement node){
    //서비스 반환을 위한 소스 코드 } }
```

(그림 11) RSEL2JAVA 알고리즘

성한 자바 프로그램은 타겟 시스템인 R-OSGi 기반 SOMAR 로봇 클라이언트에 업로드되고 로봇/디바이스 서비스 저장소를 통해 관리된다. 그리고 서비스 관리자는 원격 프로시져 콜에 따라 함수의 호출을 관리한다. (그림 11)은 원격 프로시져 호출을 위한 SOMAR 로봇 클라이언트 미들웨어의 구조를 보인다.

로봇 클라이언트에서 생성한 서비스는 URC 서버에서 프록시 클래스 (Proxy Class)로 사용할 수 있으며, 이는 URC 서버의 구조에 따라 다른 서비스로 생성할 수 있다. URC 서버는 로봇 클라이언트에서 동작하는 일부 응용을 실행하여 로봇 클라이언트를 하드웨어/소프트웨어적으로 경량화시



(그림 12) SOMAR 로봇 클라이언트 미들웨어 구조

〈표 1〉 URC 로봇 환경 비교

구 분	SOMAR	RSCA	OPRoS
소프트웨어 계층 구조	R-OSGi	RT-CORBA	프레임워크
소프트웨어 단위	서비스	컴포넌트	컴포넌트
URC 지원	지원	지원	지원
소프트웨어 호출 방식	R-OSGi Proxy Class	RT-CORBA	이진 목적 소스

킨다. <표 1>은 제안하는 로봇 클라이언트 미들웨어와 기존 URC 로봇 환경의 비교를 보이고 있다[4, 11].

<표 1>은 로봇의 실행환경을 제공하는 대표적인 연구와 본 논문에서 제안한 SOMAR를 비교하고 있다. SOMAR는 다른 환경에 비해 경량화되어 있는 R-OSGi를 기반 환경으로 사용하고 있으며 소프트웨어 단위로 서비스를 이용하고 있으나 서비스를 위한 엔진을 사용하지 않는 목적 프로그램을 생성하는 방식을 사용하고 있어 로봇 클라이언트가 경량화되어 있다.

6. 결 론

URC 로봇은 네트워크 기반의 서버-클라이언트 구조를 가진 지능형 로봇이다. URC 로봇은 클라이언트 로봇의 응용 중에서 일부를 서버에서 실행시킴으로써 클라이언트 로봇을 하드웨어/소프트웨어적으로 경량화할 수 있고, 따라서 생산원가를 절감시키고 단일 제어 서버를 통해 다수의 로봇에게 서비스를 제공할 수 있다.

본 논문은 URC 서버 환경에서 사용 가능한 SOMAR 로봇 클라이언트 미들웨어를 제안하였다. SOMAR 로봇 클라이언트는 디바이스 제어를 서비스로 정의하고 다수의 디바이스 서비스와 디바이스 응용 서비스를 결합하여 로봇 서비스를 생성한다. 또한 서비스의 결합을 직관적으로 표현하기 위해 RSEL을 제안하였다. RSEL 문서는 다수의 서비스를 조합하여 새로운 서비스를 생성할 수 있는 방법을 제공한다. 이는 디바이스 프로그램과 로봇 응용 프로그램의 독립성을 보장하고 로봇이 제공하는 서비스를 추상적으로 표현할 수 있어 효과적인 로봇 클라이언트를 개발할 수 있는 장점을 가진다. RSEL 문서로 작성한 로봇 서비스와 디바이스 서비스는 클라이언트 시스템 언어로 변환하여 원격 프로시져 호출로 URC 서버에서 사용할 수 있도록 제공한다. 이는 로봇 클라이언트에 RSEL을 처리하는 엔진을 배제하여 경량의 로봇 클라이언트를 유지할 수 있다. 또한 호스트-타겟 구조의 내장형 시스템 개발 방법에도 쉽게 적용할 수 있다.

참 고 문 헌

[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] 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.
[4] 홍성수, “RSCA : 분산 로봇 플랫폼에서 임베디드 소프트웨어의 동적 재구성을 지원하는 통합 미들웨어”, 한국통신학회지(정보통신), 제21권, 제10호, pp.22-35, 2006. 6.
[5] 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 Frist International Conference on Ubiquitous Robots and Ambient Intelligence (ICURAI), pp.426-437, December, 2004.
[6] Lorenzo Fluckiger, Vinh To, K. Kotoku, “Service Oriented Robotic Architecture Supporting a Lunar Analog Test”, International Symposium on Artificial Intelligence Robotics and Automation in Space, 2008.
[7] 곽동규, 최종선, 최재영, 유재우, “상황인지 워크플로우와 서비스 지향 미들웨어를 이용한 URC 로봇 소프트웨어 아키텍처”, 로봇학회논문지 제5권 제3호 통권17호, pp.240-250, 2010. 9.
[8] 김수연, 황석찬, 곽동규, 최재영, “URC 로봇을 위한 서비스 지향적 서버-클라이언트 미들웨어 아키텍처 설계”, 한국정보과학회 HPC 연구회 동계 학술발표대회, pp.21-26, 2009. 2.
[9] 손은미, 곽동규, 황석찬, 최재영, “URC 로봇 클라이언트를 위한 서비스 지향적 디바이스 아키텍처 설계”, 한국정보과학회 HPC 연구회 동계학술발표대회, pp.121-128, 2009. 2.
[10] 최종선, 조용윤, 최재영, “다중-워크플로우를 지원하는 상황인지 워크플로우 언어의 설계”, 한국인터넷정보학회 논문지 제10권 제6권, pp.145-157, 2009. 12.
[11] OPRoS, http://www.opros.or.kr.
[12] T. W. Fong, M. Bualat, L.Edwards, L. Flückiger, C. Kunz, S. Y. Lee, E. Park, V. To, H. Utz, N. Ackner, N. Armstrong-Crews, J. Gannon, “Human-robot site survey and sampling for space exploration”, AIAA Space 2006, September, 2006.
[13] Scott de Deugd, Randy Carroll, Kevin E. Kelly, Bill Millett, Jeffrey Richker, “SODA : Service-Oriented Device Architecture”, IEEE Pervasive Computing Vol.5, Issue 3, pp.94-96, July-Sept., 2006.
[14] J. S. Rellermeyer, G. Alonso, T. Roscoe, “R-OSGi : Distributed Applications Through Software Modularization”, Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference, 2007.
[15] OSGi, http://www.osgi.org/Main/HomePage.
[16] Bertrand Meyer, Karine Arnout, “Componentization: the Visitor example”, Computer (IEEE), Vol.39, No.7, July 2006, pp.23-30.

곽 동 규

e-mail : coolman@ss.ssu.ac.kr
2002년 서경대학교 응용수학과(학사)
2004년 숭실대학교 컴퓨터학과(공학석사)
현 재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 프로그래밍 언어, 컴파일러, XML, 임베디드 시스템, 유비쿼터스

최 재 영

e-mail : choi@ssu.ac.kr
1984년 서울대학교 제어계측공학과(학사)
1986년 미국 남가주대학교 전기·전자공학과 (공학석사)
1991년 미국 코넬대학교 전기·전자공학과 (공학박사)
1992년～1994년 미국 국립오크리지연구소 연구원
1994년～1995년 미국 테네시 주립대학교 연구교수
1995년～현 재 숭실대학교 정보과학대학 컴퓨터학부 교수
관심분야 : 시스템소프트웨어, 병렬/분산처리, 고성능컴퓨팅(HPC)