서버 측 로봇 미들웨어에서의 로봇 서비스 응용 및 제공을 위한 방법

백경윤*, 최종선*, 최재영*
*숭실대학교 컴퓨터학부
e-mail : kybaek@ss.ssu.ac.kr

A Method for applying and providing robot service in server-side robot middleware

Kyoungyun Baek*, Jongsun Choi*, Jaeyoung Choi*
*School of Computer Science and Engineering, Soongsil University

요 약

최근 로봇의 통합적인 개발과 관리 및 응용을 위한 로봇 미들웨어가 많이 연구되고 있다. 이러한 로봇 미들웨어의 로봇 서비스는 클라이언트 측에서의 로봇 서비스와 서버 측의 로봇 서비스로 구별될 수 있다. 제안하는 로봇 미들웨어 구조에서는 클라이언트 측의 로봇 서비스와 서버 측의 로봇 서비스를 구별한다. 특히 서버 측에서 클라이언트 측에서 만들어진 로봇 서비스를 응용하기 위해 로봇 서비스 실행 언어인 RSEL 을 사용하며 이를 통해 새로운 로봇 서비스를 조합하여 새로운 로봇 서비스를 만들어내고 제공한다. 제안하는 구성요소의 구현을 보이기 위해 시나리오를 이용한 시뮬레이션을 수행한다. 이 시뮬레이션을 통해 RSEL 의 문서의 작성과 그 사용을 봄으로써 제안하는 로봇 미들웨어 구조를 확인한다.

1. 서론

최근 로봇의 통합적인 개발과 관리 및 응용을 목적으로 한 로봇 미들웨어가 많이 연구되고 있다. 로봇 미들웨어는 운영체제의 상위에서 구동되는 소프트웨어로서 로봇 프로그램 및 로봇 서비스의 개발과 배포 등을 쉽게 하도록 하는 도구와 라이브러리 등을 포함한다 [1]. 로봇 미들웨어는 그 목적을 쉽고 간단하게 이루기 위해 대부분 서버-클라이언트 구조를 채용하고 있다 [2][3][4][5]. 이러한 서버-클라이언트 구조를 가진 로봇 미들웨어에서는 연결되어 있는 클라이언트 로봇에서 동작하는 로봇 서비스를 사용자에게 제공하고, 로봇 서비스들을 조합 및 응용하여 로봇 서비스를 확장할 수 있다. 로봇 서비스는 일반적으로 로봇 자신의 상태를 변경시키거나 어느 특정한 행동을 수행하는 서비스를 말하는데, 사용자는 이 서비스를 실행함으로써 로봇을 움직이거나 상태를 변경할 수 있다.

서버-클라이언트 구조의 로봇 미들웨어에서 로봇 서비스 생성은 크게 클라이언트 측에서의 생성과 서버 측에서의 생성으로 볼 수 있다. 로봇 미들웨어에서의 로봇 서비스 생성은 주로 로봇의 디바이스 드라이버를 이용하여 개발하게 되며 이에 직접 접근함으로써 이룰 수 있다. 하지만 로봇 미들웨어는 서버 측과 클라이언트 측으로 나뉘게 되며, 이 두 환경에서의 로봇 서비스 개발은 서로 다를 수밖에 없다. 특히 서버 측 로봇 미들웨어에서의 로봇 서비스 개발은 디바이스에 직접 연결되어 있는 구조가 아니기 때문에 다른 방법을 통해 로봇 서비스를 만들 수 있어야 한다.

그 동안 로봇의 움직임을 위해 많은 로봇 프로그래밍 언어들이 발표되었다. 이 로봇 프로그래밍 언어들은 각각 목적에 따라 로봇의 제어, 작업 정의, 수행, 응용, 배포, 원격 호출, 상호 소통, 프로그래밍 등의 관점으로 만들어졌다. 이 로봇을 위한 언어들은 각각의 문법과 형태를 가지고 있기도 하지만, 많은 언어들이 태그의 추가를 통해 구조를 마음대로 정할 수 있는 XML (Extensible Markup Language) 를 기반 언어로 많이 채용하고 있다. 본 논문에서 제안하는 시스템 구조에서 사용하는 로봇 서비스 조합 언어인 RSEL (Robot Service Execution Language) 또한 XML 을 기반으로 하고 있다.

따라서 본 논문에서는 XML 기반 언어인 RSEL 을 통해 서버쪽 로봇 미들웨어에서 로봇 서비스를 쉽게 조합하고 배포할 수 있는 방법과 로봇 미들웨어 구조에 대해 설명한다.

2 장에서는 그 동안 있었던 로봇 미들웨어에 대해서 설명하며, 3 장에서는 본 논문에서 제안하는 RSEL

이 논문은 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음 (NIPA-2011-C1090-1121-0010)

을 사용한 서버 쪽 로봇 미들웨어의 구조에 대해서 설명한다. 4 장에서는 로봇 서비스의 조합과 배포를 위해 필요한 시나리오와 환경, 그 실행을 보인다. 마지막으로 5 장에서는 결론에 대해 이야기한다.

2. 관련 연구

최근 세계 각지에서 로봇 미들웨어에 대한 연구가 끊임없이 이루어지고 있다. 이는 크게 미국, 유럽, 일본, 한국으로 나눌 수 있는데, 각지에서 만들어진 로봇 미들웨어는 다음과 같다.

MSRS (MicroSoft Robotics Studio) [6] 는 미국 마이크로소프트사에서 만든 .Net 기반의 로봇 응용 개발 환경 및 실행환경이다. MSRS 상에서 개발되는 로봇은 기본적으로 서비스의 조합을 통해 이루어지며, 시각적 개발 환경 지원을 통해 단순한 프로그래밍 작업을 가능하게 하는 등 개발을 손쉽도록 한다.

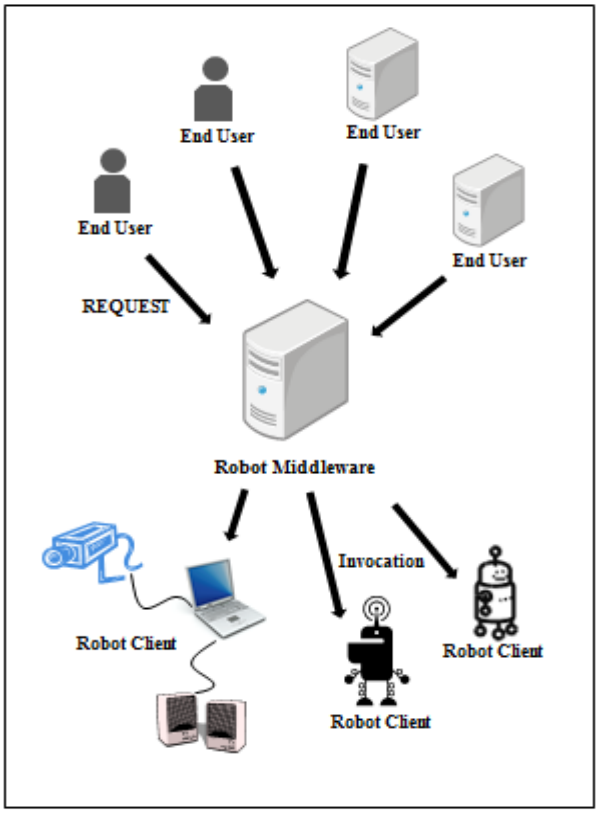
OROCOS (Open RObot Control Software)[5] 는 유럽에서 시작된 오픈 소스 프로젝트로서, 로봇의 제어를 위한 소프트웨어 개발이 그 목적이다. OROCOS 는 실시간 로봇 컴포넌트가 가지고 있어야 하는 로봇 컴포넌트와 그 인터페이스를 정의하고 있다. 이러한 컴포넌트를 쉽게 제작하고 응용할 수 있도록 실시간 툴킷 (Realtime Toolkit) 과 스크립트 언어를 제공한다.

openRTM [7]은 2008 년 일본의 AIST (National Institute of Advanced Industrial Science and Technology) 에서 RT-Middleware 의 명세를 구현한 것으로써, 컴포넌트 결합지향성 로봇 미들웨어이다. RT-Middlware 는 OMG 에서 로봇 소프트웨어 플랫폼의 단일화와 발전을 위해 제정한 표준 로봇 소프트웨어 플랫폼으로써 로봇 서비스를 컴포넌트로 구현하고 이를 서로 통신하게 함으로써 로봇을 구동하도록 한다.

OPRoS (Open Platform for Robotic Services) [8] 는 한국전자통신연구원에서 이전에 존재하던 표준 로봇 소프트웨어 플랫폼 표준인 RUPI 를 개량하여 만든 로봇 소프트웨어 플랫폼 규격이다 OPRoS 는 크게 OPRoS 서버와 OPRoS 통신 프로토콜, OPRoS 로봇 클라이언트, 로봇 콘텐츠, 응용 컴포넌트 통합 개발 환경 등으로 구성되어 있다. OPRoS 에서 로봇 응용을 하기 위해서는 OPRoS 플러그인이나 OPRoS 컴포넌트 방식으로 로봇 응용을 하게 되는 데, 이 두 방식에서는 개발자가 직접 플러그인이나 컴포넌트를 생성하여 프로그래밍을 해야 한다.

위의 개발 환경 및 플랫폼은 모두 로봇 서비스에 대한 개발 및 응용, 배포를 지원한다. 하지만 로봇 서비스를 서버 측과 클라이언트 측으로 나누어 고려하고 있지 않으며 로봇 서비스 개발 시 관련 된 컴포넌트의 재시작 등 추가적인 작업이 필요하다.

과거부터 XML 을 이용하여 로봇 서비스를 응용하는 방법에 대한 많은 연구가 있었다. RoboML [9]은 XML 을 기반으로 한 에이전트 커뮤니케이션 언어인 KQML[10]을 이용한 언어로써 로봇 서비스를 으용하고 인터넷을 통해 로봇 서비스를 사용할 수 있도록 하는 언어이다. RoboML 은 Embedded Agent 와 Interface Agent, Proxy Agent 를 이용하여 사용자가 인터넷을 통해 조합된 로봇 서비스를 이용할 수 있도록 한다. 하지만 기본적인 스키마가 단순하기 때문에 로봇 서비스의 단순한 조합만 가능할 뿐 그 확장에 대해서는 크게 기대하기 힘들다.



(그림 1) 제안하는 로봇 미들웨어의 구조

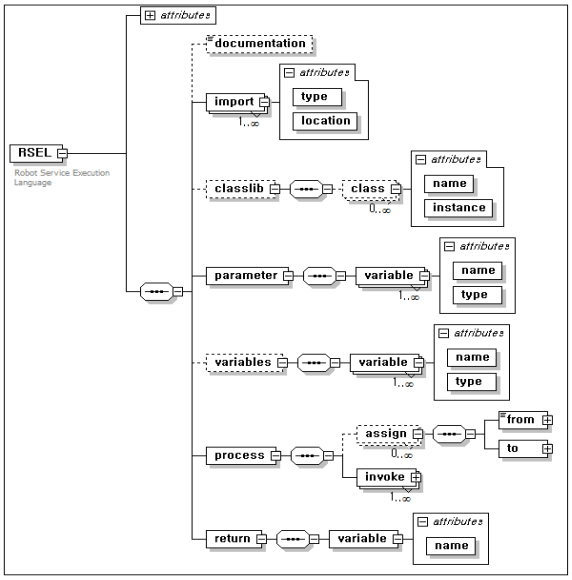
3. 제안하는 미들웨어 구조 및 RSEL

본 논문에서 제안하는 로봇 미들웨어의 구조는 그림 1 과 같은 환경에서 적용될 수 있다. 이 환경에서는 크게 세 가지 분류의 집단이 존재한다. 한 가지는 로봇 서비스를 실행하는 사용자이거나 혹은 로봇 서비스를 통해 다른 어떤 응용 서비스를 개발하는 시스템이며, 다른 한 가지는 한 대 이상의 클라이언트 로봇이 연결되어 클라이언트 로봇을 관리하고, 클라이언트 로봇이 가지고 있는 로봇 서비스를 배포하는 서버 측 로봇 미들웨어이다. 마지막 한 가지는 사용자가 선택한 로봇 서비스가 실제로 수행되는 클라이언트 로봇이다. 종합해서 말해보면, 로봇 미들웨어는 로봇 클라이언트들이 제공하는 로봇 서비스를 사용자 혹은 시스템에 제공한다. 본 구조는 클라이언트 로봇의 서비스를 한 곳으로 집중하여 관리하고 배포함으로써 사용자가 사용하기 쉽고 클라이언트 로봇을 관리하기 쉽다는 장점이 있다.

3.1 RSEL

RSEL (Robot Service Execution Language)은 XML 기반의 언어로써, 로봇 미들웨어에서 로봇 서비스를 조합하여 새로운 로봇 서비스를 만들고 배포하기 위한 언어이다. 서버 쪽의 로봇 미들웨어에서 개발자는 클

라이언트 로봇의 디바이스 드라이버에 직접 접근할 수 없기 때문에, 클라이언트 로봇에서 제공하는 로봇 서비스를 통해 새로운 서비스를 만들어야 한다.

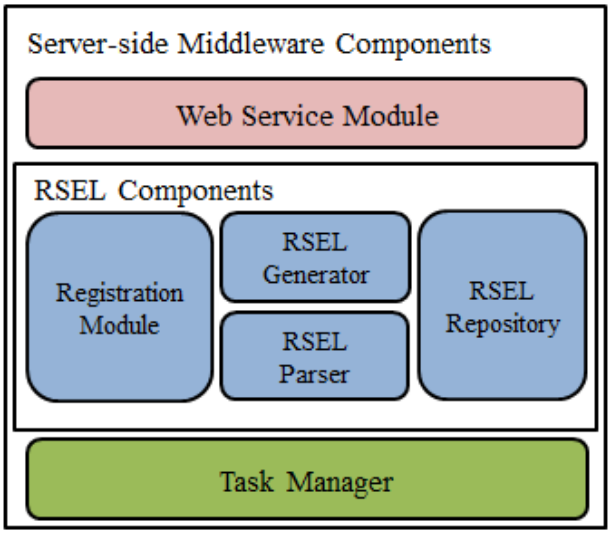


(그림 2) RSEL 스키마

RSEL 의 스키마는 비즈니스 프로세스를 XML 언어를 통해 정의하고 그것을 워크플로우로써 실행시키는 언어인 BPEL (Business Process Execution Language) 의 스키마를 기본으로 하고 있다. 그렇기 때문에, 두 언어 스키마 기본적인 형태와 가지고 있는 태그가 유사하다. 하지만 RSEL 에 추가적인 태그가 있기 때문에, 호환은 되지 않으며, 그 역할 또한 매우 다르다. 그림 2 는 RSEL 의 스키마이다. RSEL 스키마에 대한 세부적인 설명은 다음과 같다. 'documentation' 태그는 서비스에 대한 내용을 기술한다. 'import' 태그는 현재 작성되는 서비스가 참조하는 WSDL 의 위치를 가리킨다. 'type' 태그는 서비스의 종류를 가리킨다. 'location' 태그는 WSDL 의 URI 혹은 URL 을 가리키킨다.'classlib' 태그는 디바이스 서비스와 로봇 서비스를 위한 라이브러리를 가리킨다. 'parameter' 태그는 서비스 내에서 서브 로봇 서비스들에 input 으로 사용되는 변수를 나타낸다. 'variable' 태그는 서비스 내부에서 사용되는 변수를 나타낸다. 'process' 태그는 assign 과 invoke 태그를 가지고 있는데, 'assign' 태그는 변수들 사이의 계산이나 배정을 수행하며, 'invoke'태그는 하위의 서비스를 실행시킨다. 'return' 태그는 서비스에서 리턴하는 값을 정의한다. 이 'return' 태그는 RSEL 내부의 paramter 나 variable 이 될 수 있다.

3.2 RSEL 컴포넌트

RSEL 을 사용하기 위해서는 그림과 같은 컴포넌트들이 서버 측 미들웨어에 존재하여야 한다. 가지고 있어야 하는 컴포넌트에는 컴포넌트에는 RSEL Parser, RSEL Registration Module, RSEL Repository, RSEL



(그림 3) RSEL Components

Generator 가 있으며 효과적인 서브 로봇 서비스들의 수행을 위해 Task Manager 와 로봇 서비스 배포를 위한 Web Service Module 이 필요하다. 각각의 컴포넌트들은 RSEL 의 사용을 위해 일정한 역할을 수행하는데, 그 역할은 다음과 같다. RSEL Registration Module 은 RSEL 문서를 등록하기 위해 필요하며, 등록된 RSEL 문서는 RSEL Parser 를 거쳐 필요한 정보 수집 후 RSEL Repository 에 저장된다. RSEL Genorator 는 차후에 RSEL 문서를 더 쉽게 제작하기 위해 필요한 정보만을 넣고 문서생성을 자동으로 하기 위한 모듈이다. 이렇게 RSEL Repository 에 저장 된 정보는 Web Service Module 을 통해 사용자에게 제공되고, Task Manager 를 통해 클라이언트 로봇의 로봇 서비스를 수행하게 된다.

4. 시나리오 시뮬레이션

시나리오 시뮬레이션에서는 RSEL 을 통해 로봇 서비스를 만들고 배포할 수 있는 시나리오를 작성하고 이 시나리오에 따라 RSEL 문서를 작성하여 본다. 작성된 RSEL 문서는 RSEL Registration Module 과 Web Service Engine 을 통해 사용자에게 제공될 수 있다.

서버 측 로봇 미들웨어에서 클라이언트 로봇과 연결하여 서비스를 제공하고자 하는 시나리오는 다음과 같다.

> 한 행사장에 안내 로봇이 있다. 이 안내 로봇은 방문객이 자신이 가지고 있는 모바일 기기를 통해 서비스를 실행하게 되면, 안내 로봇이 방문자에게 다가온다. 그 후 로봇이 전시장의 지도를 보여주고 사용자가 입력을 할 때까지 기다린다. 사용자가 행사장 한 부스의 위치를 입력하거나 어느 한 부스를 선택하게 되면 로봇이 선택된 부스로 방문객을 인도한다. 부스에 도착하고 난 뒤 로봇은 해당 부스에 대한 전시 자료를 보여주고, 이에 대해 브리핑을 한다.

(그림 4) 시뮬레이션 시나리오

위 시나리오를 수행하기 위해서는 안내 로봇에는 기본적인 로봇서비스가 존재해야 한다. 서버 측 로봇 미들웨어에서의 로봇 서비스 조합은 클라이언트 로봇에서 제공하는 로봇 서비스들을 바탕으로 이루어질 수 있기 때문에 클라이언트 로봇인 안내 로봇에서 필요한 로봇 서비스를 제공해야한다. 다음 표는 안내 로봇에서 제공해야 하는 로봇서비스의 목록이다.

<표 1> 안내 로봇이 가지는 로봇 서비스

	Service Name	Arguments
1	move	-
2	moveToUser	User location
3	moveToBooth	Booth Number
4	displayInformation	Booth Number
5	briefInformation	Booth Number
…	…	…

위의 로봇 서비스를 통해 서버 측 로봇 서비스 개발자는 그림 5 와 같은 RSEL 문서를 작성하여 새로운 로봇 서비스를 개발할 수 있다. 이 조합된 로봇 서비스는 사용자에게 순차적으로 세부 서비스들을 제공함으로써 사용자가 각각 수행해야 했던 불편함을 없애줄 수 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<RSEL layer="robotService" name="Boothguide"
  …
    <parameter>
        <variable name="$userLocation"/>
        <variable name="$boothNumber" />
            <variable name="$processResult" />
    </parameter>
    <process>
        <!– send data –>
        <invoke layer="robotService"
                operation="moveToUser"
                    inputVariable="$userLocation"
                    outputVariable="$processResult"/>
        <invoke layer="robotService"
                operation="moveToBooth"
                    inputVariable="$boothNumber"
                    outputVariable="$processResult"/>
        <invoke layer="robotService"
                    operation="displayInformation"
                    inputVariable="$boothNumber"
                    outputVariable="$processResult" />
        <invoke alyer="robotService"
                    operation="briefInformation"
                    inputVariable="$boothNumber"
                    outputVariable="$processResult" />
    </process>
    <return>
        <variable name="$processResult"/>
    </return>
</RSEL>
```

(그림 5) 조합된 로봇 서비스를 기술하는 RSEL 문서의 일부분

그림 5 의 RSEL 문서를 보면 invoke 태그에 표 1 에서 본 로봇 서비스들이 적혀있는 것을 볼 수 있다. 사용자가 이 가이드 로봇 서비스를 실행하게 되면 RSEL Parser 는 이 부분을 RSEL Repository 에 저장하여 두고 순차적으로 실행함으로써 전체적인 로봇 서비스가 이루어질 수 있다.

5. 결론

이 논문에서 우리는 다양한 로봇 서비스를 조합하고 재사용할 수 있게 해 주는 미들웨어 아키텍쳐에 대해 설명하였다. 또한 우리는 RSEL 을 통해 로봇 서비스를 조합하고 실행하는 과정을 보였다. 제안한 미들웨어 구조는 로봇 서비스를 조합하고 실행하기 위한 로봇 서비스 실행언어인 RSEL 에 기초하고 있다. 이에 추가적으로 안내 로봇에 대한 예시를 통해 로봇 서비스의 재사용과 조합을 제안한 미들웨어 구조가 지원한다는 것을 보였다.

앞으로의 연구에서는 RSEL 의 확장을 통해 서브 서비스들의 유연한 조합이 필요할 것이다.

참고문헌

[1] van Breemen, A.J.N, “Scripting Technology and Dynamic Script Generation for Personal Robot Platforms”, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3487-3492, 2005.

[2] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, “A Review of Middleware for Networked Robots”, International Journal of Computer Science and Network Security, Vol.9 No.5 pp. 139-148, 2009.

[3] M. Munich, J. Ostrowski, and P. Pirjanian, “ERSP : A software platform and architecture for the service robotics industry”, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 460-467, 2005

[4] RTC, “The Robotic Technology Compnent Specification”, OMG Adopted Specification, ptc/06-11-07, Nov. 2006

[5] Bruyninckx, H, “Open robot control software : the OROCOS project”, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference, vol. 3, pp. 2523-2528, 2001.

[6] MSRS, “http://www.microsoft.com/korea/robotics/”

[7] Noriaki Ando, Takashi Suehiro, Tetso Kotoku, “A Software Platform for Component Based RT-System Development : OpenRTM-Aist”, Simulation, Modeling, and Programming for Autonomous Robots, pp. 87-98, 2008.

[8] OPRoS, “http://www.opros.or.kr”

[9] Maxim Makatchev, S. K. Tso, “Human-Robot Interface Using Agents Communicating in an XML-based Markup Language”, 9th IEEE International Workshop on Robot and Human Interactive Communication, pp. 270-275, 2000.

[10] T. Finin, Y. Labrou, J.Mayfield, “KQML as an Agent Communication Language”, in Software Agents, ed. J. M. Bradshaw, AAAI Press/The MIT Press, pp. 291-316, 1997.