유비쿼터스 컴퓨팅 환경에서 로봇 자동화 서비스를 위한 계층적 아키텍처

A Layered Architecture for Robot Automated Services on Ubiquitous Computing Environments

최 종 선[1], 최 재 영[†], 조 용 윤[2]

Jongsun Choi[1], Jaeyoung Choi[†], Yongyun Cho[2]

Abstract In ubiquitous computing environments, users want to receive the robot services regardless of various physical status or devices such as time, place, various sensors, and high-performance servers. Thus, the ubiquitous service robots have to provide users with automated services according to situational information that they properly recognize. Beyond these problems, robot software has to establish a foundation to support the functions with the network infrastructure that are not able to be solved by a single independent resource. On the basis of a robot middleware that is capable of minimizing dependencies among hierarchy structures, the robot software also has to provide execution environment to control the flow of robot application services. In this paper, we propose a layered architecture to provide users with automated services through ubiquitous robots. The proposed architecture is based on CAWL (Context-Aware Workflow Language) and RSEL (Robot Services Execution Language). CAWL easily represents the flow of robot services from user application service levels, and RSEL is able to support the composition and reusability of robot services through abstraction of robot device services. In our experiments, we applied the proposed architecture to an example of “booth guide robot service.”

Keywords: Ubiquitous Computing, Ubiquitous Robot, Automated Services, Layered Architecture, CAWL, RSEL

1. 서 론

유비쿼터스 컴퓨팅과 같은 새로운 패러다임의 등장으로 로보틱스 분야에 3세대 로봇인 유비쿼터스 로봇의 개념이 새롭게 나타나기 시작했으며, 최근에는 로보틱스을 이용한 서비스를 제공하기 위한 미들웨어 및 네트워크 기반 로봇 시스템에 관한 연구들이 국내·외에서 활발히 진행되고 있다[1-6]. 이와 같은 패러다임의 변화는 로봇 산업의 초점을 산업용 로봇에서 자율 방식으로 일상 공간에서 사람에게 유용한 서비스를 제공하는 지능형 서비스 로봇 분야로 확장시키고 있다. 유비쿼터스 컴퓨팅 환경에서 사용자는 시간, 장소, 각종 센서 및 고성능 서버와 같은 물리적 디바이스에 구애 받지 않으면서 로봇 서비스를 제공 받고자 한다. 따라서 서비스를 제공하는 주체인 유비쿼터스 로봇은 다양한 주변 상황을 올바로 인식하여, 그것에 따라 사용자에게 자율적인 자동화 서비스를 제공해야 한다. 그리고 이와 같은 자동화 서비스를 실현하기 위한 모델로써 WS-BPEL와 같은 표준 워크플로우 기술을 유비쿼터스 컴퓨팅 환경에 적용해야 할 필요성이 있다. 웹 서비스 기반의 워크플로우 기술은 이질적인 하드웨어 플랫폼을 통해 제공되는 다양한 로봇 서비스를 쉽게 통합할 수 있는 수단을 제공하며, 소프트웨어 아키텍처를 구성하는데 낮은 결합도를 제공하여 서비스 모듈간 종속성을 줄여 준다.

Received : Oct. 17. 2011; Reviewed : Nov. 07. 2011; Accepted : Nov. 23. 2011
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 육성지원 사업의 연구결과로 수행되었음(NIPA-2011-C1090-1121-0010)
† 교신저자 : 숭실대학교 IT대학 컴퓨터학부 교수
[1] 숭실대학교 지능형로봇연구소(ITRC) 전임연구원
[2] 순천대학교 정보통신공학부 조교수

사용자에게 제공하는 로봇 서비스는 일반적으로 사용자에게 제공되는 최상위 수준의 응용 서비스 정의를 시작으로, 상황 정보에 따른 로봇 서비스의 관계를 정의하는 로봇 응용 개발 계층, 로봇 디바이스의 추상화를 통한 로봇 서비스를 구성하는 로봇 서비스 개발 계층, 마지막으로 최하위 수준의 하드웨어 디바이스를 운용하기 위한 로봇 디바이스 소프트웨어 계층에 이르기까지 4개의 계층으로 구성된 아키텍처를 갖는다[2,6]. 이와 같은 아키텍처에서 로봇의 자동화 서비스를 위한 기술로써 워크플로우 개념을 상위 2개 계층에 적용할 수 있다. 일반적으로 개발자들은 워크플로우 기술을 적용하기 위해 행동 관계를 정의할 수 있는 규칙 언어를 새롭게 개발하거나 BPEL과 같은 기존 워크플로우 언어를 사용하고, 이에 상황인지를 위해 JESS와 같은 추론 엔진을 추가적으로 사용함으로써 상황 정보에 따른 로봇 서비스의 관계를 정의할 수 있다. 하위 2개 계층은 로봇 디바이스를 제어하기 위한 로봇 디바이스 서비스 계층과 이 계층의 추상화를 통한 로봇 서비스 계층이며, 일반적으로 개발 계층간 종속성을 줄이기 위해 서비스 지향 구조(SOA) 기반의 소프트웨어 개발 방법론과 클라이언트-서버 형태의 구조를 갖는다[2,3].

로봇 소프트웨어는 독립적인 자원이 스스로 해결하기 어려운 기능을 네트워크 인프라를 통해 해결할 수 있는 기반 구조를 가져야 한다. 또한 개발 계층 간 의존성을 줄일 수 있는 미들웨어 구조를 지원하여, 로봇 응용 서비스의 흐름을 제어할 수 있는 실행 및 제어 환경을 제공해야 한다. 이를 위해 본 논문에서는 유비쿼터스 컴퓨팅 환경에서 로봇을 통해 사용자에게 자율적인 자동화 서비스를 제공하기 위한 계층화 아키텍처를 제안한다. 제안하는 아키텍처는 응용 서비스 수준에서 상황인지를 기반으로 로봇 서비스의 흐름 및 제어를 표현할 수 있는 CAWL[7]과 로봇 디바이스 서비스를 추상화한 로봇 서비스의 조합 및 재사용을 위한 RSEL을 활용한다. CAWL은 RDF 트리플 기반의 컨텍스트 모델을 적용하여 상황 정보를 CAWL 문서에 직접 표현할 수 있으며, RSEL은 실제 로봇 디바이스를 제어하는 디바이스 드라이버, 즉 로봇 디바이스 서비스를 추상화하여 다양한 로봇 서비스의 형태로 조합하여 RSEL 문서에 표현할 수 있다.

2. 관련 연구

IBM, Microsoft, BEA, Oracle 등과 같은 세계적인 컴퓨터 회사의 검증을 거친 비즈니스 프로세스 통합을 위한 자동화 모델인 워크플로우 기술을 유비쿼터스 컴퓨팅 환경에 적용하려는 연구들이 최근까지 이루어지고 있다[8-12]. 또한 상황인지 미들웨어의 연구 영역을 로보틱스 분야로 확장한 지능형 로봇 미들웨어에 관한 연구들이 활발히 진행되고 있다[13-18]. 본 장에서는 상기 언급한 워크플로우 기술과 미들웨어 연구에 관한 최근 동향 및 관련 기술[19-21]에 대하여 살펴보도록 한다.

2.1 상황인지 워크플로우

Context4BPEL[8]은 표준 워크플로우로 인정받고 있는 WS-BPEL[9]을 확장하여 이를 실행하기 위한 플랫폼의 구조를 제안하였으며, 이를 확장한 연구[10]에서는 스마트 워크플로우 개념을 소개하였다. Context4BPEL에서는 WS-BPEL 확장을 위한 워크플로우 스키마의 수정이 불가피하였으므로, 워크플로우 처리 엔진을 수정해야 한다. FollowMe[11]는 유비쿼터스 컴퓨팅 환경을 위한 OSGi 프레임워크로써, 유비쿼터스 컴퓨팅 환경에서 존재하는 다양한 도메인에 적용할 수 있는 상황인지 워크플로우 기반 구조를 제공하는 목적을 가지고 있으나, FollowMe에서 사용하는 워크플로우 언어 및 이를 처리하기 위한 추가적인 연구가 필요하다. uFlow[12]는 기존 웹 서비스 기반 워크플로우 언어들이 서비스 전이 조건으로 상황 정보를 기술하지 못하는 점을 해결하기 위하여 새로운 유비쿼터스 워크플로우 언어인 uWDL을 기반으로 하는 상황인지 워크플로우 서비스 프레임워크이다. uWDL은 워크플로우가 주변 상황에 대한 상황을 인지하고, 상태를 전이할 수 있도록 워크플로우의 상태 전이 제약 조건에 상황 정보를 명시하기 위한 구조적 컨텍스트 모델을 기반으로 하고 있으며, 이러한 uWDL을 통해 시나리오에 기술한 상황 정보에 따라 사용자의 서비스 요구를 처리하고 실행한다. CAWL은 uWDL의 단점을 개선하기 위해 uWDL 스키마의 전반적인 구조를 수정한 상황인지 워크플로우 언어이다. CAWL[7]에서는 uWDL을 통해 시나리오 문서 내에 상황 정보 정보를 정적으로 기술하고, 다수의 워크플로우 서비스를 표현할 수 없는 단점이 보완되었으며, 이를 통해 유비쿼터스 컴퓨팅 환경에 존재하는 다양한 u-시스템을 이용한 서비스를 포함하여 지능형

로봇 서비스까지 표현할 수 있게 되었다. 본 논문에서는 이와 같이 개선된 CAWL을 사용자 수준에서의 로봇 응용 서비스를 표현하기 위한 기반 기술로 활용한다.

2.2 상황인지 워크플로우

현재 OMG에서 로봇 미들웨어의 표준을 이끌고 있는 일본을 비롯하여 세계의 각 나라들은 네트워크 기반의 지능형 로봇을 활용하여 로봇 산업을 육성하려고 한다. 특히 우리나라 정부는 세계 최고 수준의 통신 인프라를 이용하여 로봇 기술을 고도화하려는 노력을 기울이고 있다[13]. 이 같은 노력의 일환으로 국내의 경우 ETRI에서는 2005년부터 USN 기반 Ubiquitous Robotic Space 과제를 통하여 RFID, UWB, ZigBee 등의 네트워크 환경을 적극 이용하는 지능형 서비스 로봇을 개발하고 있다[14]. ETRI에서 개발한 상황인지 기반 미들웨어인 CAMUS는 CORBA를 기반으로 개발하고 있는 URC 시스템을 위한 상황인식 미들웨어로, URC 환경 내에서 획득된 상황 정보를 기반으로 환경 내에 있는 사용자에게 적절한 서비스를 제공할 수 있도록 상황 기반 응용의 개발과 실행을 지원한다[14,15,16]. 현재 정부에서는 OPRoS[2]로봇 소프트웨어 플랫폼으로 통일된 규격을 따르도록 하고 있다. OPRoS는 한국전자통신연구원(ETRI)에서 기존 RUPI 규격에서 출발하여 CAMUS를 거쳐 수정 보완된 로봇 소프트웨어 플랫폼 규격으로서, 현재 한국은 이와 같은 기술을 바탕으로 일본과 함께 OMG에서 로봇 미들웨어 표준 활동에 동참하고 있다. OPRoS는 크게 OPRoS 서버, OPRoS 통신 프로토콜, OPRoS 로봇 클라이언트, 로봇 콘텐츠 그리고 응용 컴포넌트 통합 개발 환경 등으로 구성된다. OPRoS의 로봇 응용은 OPRoS 플러그인 또는 OPRoS 컴포넌트 방식을 적용하는데, 개발자는 이 두 가지 방식을 바탕으로 플러그 인이나 컴포넌트를 생성하는 프로그래밍을 작성해야 한다.

국외의 경우는 일본의 산업기술종합연구소(AIST)에서 진행 중인 OpenRTM-aist[3]가 있으며, 2006년 표준으로 채택되었다[17]. 각각의 로봇 시스템이 가지고 있는 특수성 때문에 로봇 시스템을 지원하는 소프트웨어의 개발은 재사용성이 떨어지는 문제점을 갖는다. 이를 해결하기 위해 RT-미들웨어는 네트워크 기반의 다양한 로봇 디바이스들에 소프트웨어의 모듈화 개념을 적용하여 컴포넌트의 재사용성을 향상시켰다. 그 외에도 서비스 지향 구조(SOA) 기반의 로봇 소프트웨어 개발 방법론을 적용한 대표적인 미들웨어로 SORA[18]가 있으며, 이는 미국 NASA에서 개발한 로봇을 위한 소프트웨어 구조이다. 또한 BPEL과 같은 표준 워크플로우 기술을 적용한 로봇 미들웨어로 Tavetanov가 제안한 로봇 시스템[19]이 있으며, 이 시스템은 로봇의 프로세스를 기술하는 BPEL 워크플로우 언어 자체가 상황 정보를 표현하는데 어려움이 있으므로, BPEL를 수정하여 상황인지 기능을 추가하거나 별도의 추론 엔진을 사용해야 한다.

2.3 XML 기반의 로봇 서비스 응용 방법

로봇 미들웨어는 로봇 시스템을 제어하기 위해 일반적으로 계층적인 구조의 미들웨어가 필요하다. 이와 같은 로봇 미들웨어의 최하위 계층은 로봇 디바이스가 있고, 이를 제어하기 위한 상위 계층으로 다소간 명칭의 차이는 있지만 일반적으로 로봇 응용 계층이 존재한다. 특히 이 로봇 응용 계층의 설계를 위해 10여년 전부터 XML을 활용한 로봇 서비스 응용 방법에 대한 연구들이 진행되어 왔다. 구체적인 내용은 다음과 같다. RoboML[19]은 XML을 기반으로 한 로봇 서비스 응용을 위한 대표적인 예로 에이전트 커뮤니케이션의 목적을 가진 KQML[20]을 바탕으로 한다. 또한 RoboML은 로봇 서비스를 응용하기 위한 방법을 제공하며, Embedded Agent, Interface Agent, 그리고 Proxy Agent를 이용하여 사용자가 조합된 로봇 서비스를 인터넷을 통해 이용할 수 있는 기반을 제공한다. 그러나 다양한 로봇 서비스를 지원하기 위한 상위 수준의 로봇 서비스를 위한 서비스 조합 기능을 제공하는데 어려움이 있다. 캠프리지 대학에서 만든 CURL (Cambridge University Robot Language)[21]은 사람을 지향하는 프로그래밍에 초점을 둔 로봇 시스템을 위한 프로그래밍 언어이다. CURL의 문법은 사람의 언어 형태와 유사하여 CURL을 사용하는 개발자가 쉽게 이용할 수 있다. CURL은 재활 보조 시스템 제어를 위한 도구로 활용되고 있으나, 로봇 응용 서비스를 개발하지 위한 수단으로는 적합하지 않다.

이상 살펴본 바와 같이 유비쿼터스 컴퓨팅 환경에서의 다양한 서비스를 제공하기 위한 노력들이 서비스의 자동화를 위한 워크플로우 기술, 지능형 로봇 서비스를 제공하기 위한 로봇 미들웨어 기술, 그리고 로봇 시스템을 위한 XML 기반의 응용 기술 등의 다양한 형태로 이루어지고 있다. 본 논문에서도 상기 언급한 기술 내용을 주축으로 삼는다. 상황인지 기반 로봇 서비스의 자동화를 위해

상황 정보를 표현할 수 있는 워크플로우 기술을 반영한 CAWL과 로봇 디바이스 서비스의 추상화를 통해 로봇 서비스를 조합하여 보다 복잡하고 다양한 상위 서비스로의 확장성을 제공하는 RSEL을 제안하는 소프트웨어 아키텍처에 적용하였다.

3. 계층적 로봇 소프트웨어 아키텍처

유비쿼터스 컴퓨팅 환경에서 사용자에게 상황인지 기반의 로봇 서비스를 제공하기 위해서는 다양한 서비스를 표현하고 처리할 수 있는 컴퓨팅 환경이 필요하다. 본 논문에서는 여러 워크플로우의 흐름을 하나의 흐름으로 통합하여, 보다 큰 규모의 복합 워크플로우 서비스를 제공할 수 있는 CAWL과 로봇 디바이스 서비스의 추상화를 통한 다양한 로봇 서비스를 응용할 수 있는 기능을 제공하는 RSEL에 기반한 계층적 로봇 소프트웨어 아키텍처를 제안한다.

제안하는 아키텍처를 크게 보면 관련 연구에서 언급한 CAWL과 RSEL을 핵심 컴포넌트로 가지고 있으며, 그 외에 핵심 컴포넌트를 처리하기 위한 엔진들로 구성된다. 제안하는 아키텍처의 개요는 다음 그림과 같다.

그림 1에서 보는 바와 같이 제안하는 로봇 소프트웨어 아키텍처는 다음과 같은 4개의 계층으로 구성된다.

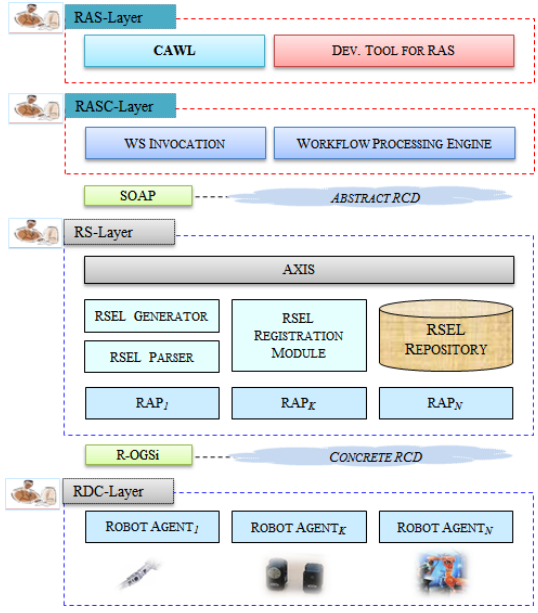

그림 1. 제안하는 로봇 소프트웨어 아키텍처 개요

- RAS-Layer: 로봇 응용 서비스 계층 (Robot Application Services)
- RASC-Layer: RAS 제어 계층 (RAS Control)
- RS-Layer: 로봇 서비스 계층 (Robot Services)
- RDC-Layer: 로봇 디바이스 제어 계층 (Robot Device Control)

3.1 로봇 응용 서비스 계층 (RAS-Layer)

RAS 계층은 제안하는 소프트웨어 아키텍처의 최상위 계층으로 사용자 수준에서의 로봇 서비스를 정의한다. CAWL[10]은 사용자 수준의 응용 서비스를 상황 정보를 바탕으로 기술할 수 있는 상황인지 워크플로우 언어로, 본 논문에서는 CAWL을 로봇 서비스를 기술하는데 사용한다. 다음 그림은 CAWL의 개념도이다.

그림 2에서와 같이 CAWL은 웹 서비스를 기반으로 사용자 및 디바이스에 대한 프로파일, 위치, 시간 등과 같은 상황 정보를 가지고 사용자에게 적합한 서비스를 선택할 수 있는 기능을 제공할 수 있으며, 다수의 사용자를 위해 워크플로우 서비스의 흐름을 제어하고 동적인 상황 정보를 표현할 수 있는 상황인지 기반의 워크플로우 언어이다. 서비스 개발자는 여러 사용자를 위한 다중-워크플로우 서비스를 지원하기 위해 다수의 CAWL을 작성할 수 있으며, 상황 정보는 컨텍스트 모델[16]을 기반으로 기술한다. 이때 워크플로우 시나리오, 즉 로봇 서비스는 개발자가 직접 기술할 수도 있고, 로봇 응용 서비스 개발을 위한 개발 도구를 이용하여 직관적인 UI를 통해 로봇 워크플로우 서비스를 구성할 수도 있다.

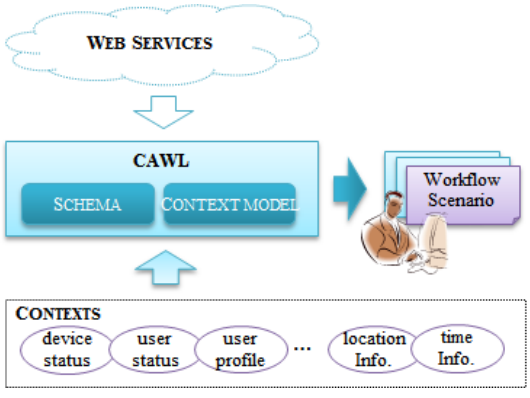

그림 2. CAWL의 개념도

3.2 로봇 응용 서비스 제어 계층 (RASC-Layer)

RASC 계층은 제안하는 소프트웨어 아키텍처의 두 번째 계층으로 상위 계층(RAS)에서 정의한 워크플로우 서비스의 흐름을 제어하고 및 각각의 로봇 서비스들을 호출하여 실행시키는 역할을 수행한다. 그림 3은 RASC 계층의 세부 구조를 나타낸다. 그림 3에서와 같이 RASC 계층은 상황 정보를 표현하기 위한 RDF 기반의 구조적 컨텍스트 모델을 기반으로 삼고 있으며, 이를 바탕으로 로봇 서비스(RS)를 호출한다. 또한 RASC 계층은 사용자 수준의 로봇 응용 서비스를 정의한 CAWL 문서를 입력 값으로 사용하며, 이 문서를 바탕으로 상황 정보에 따른 로봇 서비스 흐름의 의미를 분석하여 사용자에게 적합한 로봇 서비스를 제공한다. 각각의 구성 요소를 살펴보면 다음과 같다.

CAWL Parser는 CAWL 이용하여 작성한 시나리오 문서를 파싱하는 기능을 수행한다. 이를 통해 로봇 서비스 실행에 필요한 다양한 변수 또는 워크플로우 관련 정보를 시나리오 문서로부터 추출하여, 변수는 심볼 테이블을 통해 관리하고 워크플로우 정보는 AST(Abstract Syntax Tree) 형태로 유지한다. 이렇게 저장된 변수는 워크플로우 서비스의 실행을 위한 정보로 사용되며, 워크플로우의 AST 정보는 동일한 서비스에 대한 재사용성을 제공하는데 이용할 수 있다. 유비쿼터스 컴퓨팅 환경에서 사용자의 상황 정보에 적합한 서비스를 제공하기 위해서는 센서에서 발생하는 상황 정보와 시나리오 문서에 기술한 상황 정보를 비교 및 판단할 필요가 있다. 이를 위해 RASC 계층에서는 컨텍스트 비교기를 제공한다. 컨텍스트 비교기의 역할은 AST의 상황 정보, 즉 시나리오 문서에 RDF 형식으로 기술한 상황 정보와 센서로부터 들어온 컨텍스트 정보를 구조적 컨텍스트 모델을 기반으로 가공한 상황 정보가 일치하는지를 검증한다. 이와 같이 컨텍스트 정보를 바탕으로 사용자의 상황을 판단한 CAWL 실행 엔진은 상황에 적합한 서비스, 즉 웹 서비스(WS)를 호출한다. 호출되는 웹 서비스는 하위 계층(RS-Layer)에서 배포한다.

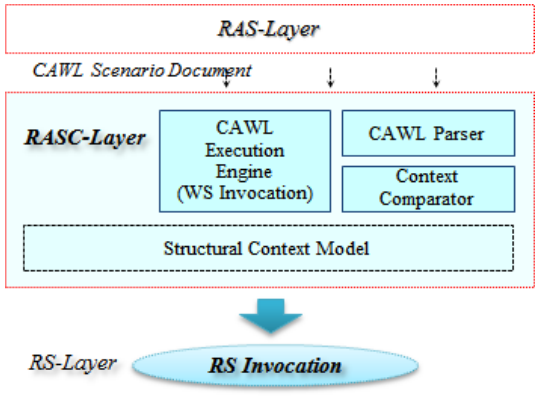

그림 3. RASC 계층의 구조 및 역할

유비쿼터스 컴퓨팅 환경에서 사용자의 상황정보에 적합한 로봇 자동화 서비스를 제공하기 위해서는 주변 상황을 판단할 수 있어야 한다. 이를 위해 제안하는 아키텍처의 RASC 계층에서는 센서로부터 얻을 수 있는 상황정보와 CAWL을 이용하여 시나리오 문서에 기술한 상황정보를 비교 및 판단할 수 있는 컨텍스트 비교기를 제공한다. 컨텍스트 비교기의 역할은 AST의 상황정보, 즉 시나리오 문서에 RDF 트리트플 형식으로 기술한 상황정보와 센서로부터 획득한 컨텍스트 정보를 구조적 컨텍스트 모델(structual context model)[12]을 기반으로 가공한 상황정보가 일치하는지 검증하는 것이다. 구조적 컨텍스트 모델에 관한 상세한 내용은 본 연구진이 2006년에 개발한 uFlow[12]에서 확인할 수 있다.

그림 4는 센서로부터 획득하여 구조적 컨텍스트 모델을 거쳐 주어, 동사, 목적어 형태의 상황정보와 응용 서비스 개발자가 CAWL 시나리오 문서상에서 기술한 상황정보를 비교하는 과정을 나타낸다. 그림 4에서 볼 수 있듯이 사용자 상황정보 및 시나리오에서 표현되는 컨텍스트는 RDF 기반 {주어, 동사, 목적어} 형태의 엔티티(entity)들에 대한 집합으로 표현된다. 즉, 센서 네트워크로부터 발생하는 컨텍스트 정보는 RDF 기반의 구조적 컨텍스트 모델을 통해 {주어, 동사, 목적어}로 구성된 엔티티의 집합으로 객체화될 수 있다.

그림 4에서 컨텍스트 비교 모듈의 입력 정보는 그림 4의 상단의 시나리오 정보와 하단의 객체화된 컨텍스트 (OC: Objectified Context) 정보이다. 예를 들어 표현한 시나리오의 내용은 “Person에 해당하는 사람이 OfficeRoomNum에 위치하는가?”를 판단하는 내용을 기술한 것이다. 이때 Person과 OfficeRoomNum의 정보는 문서에 수치로 값이 정해져 있는 정적인 정보가 아니고, 센서로부터 전달받은 값, 즉 사람과 사무실에 대한 정보를 판단한 후 입력되는 정보이다. 이렇게 입력된 정보는 그림 3의 CAWL 파서를 통해 문서 인스턴스 AST를 생성하며, 이 정보는 컨텍스트 비교 모듈의 입력 값이 된다. 그리고 그림 4에서 하단의 입력 값은 실제 센싱된 사람의 정보와 사무실 방 번

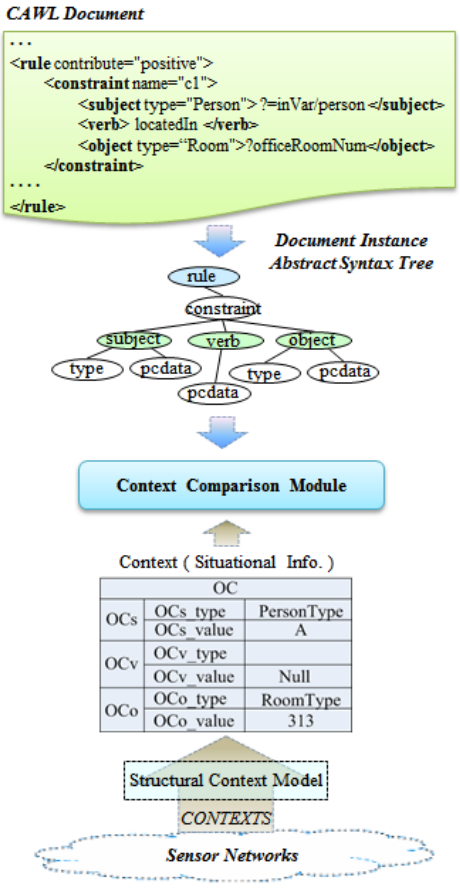

그림 4. 상황정보의 추출 및 비교 과정

호에 대한 컨텍스트 정보를 구조적 컨텍스트 모델을 통해 객체화시킨 정보이다. 따라서 컨텍스트 비교 모듈은 이 두 입력 값의 비교를 통해 사용자의 주변상황을 판단할 수 있다.

3.3 로봇 서비스 계층과 로봇 디바이스 제어 계층

로봇 서비스(RS) 계층은 제안하는 소프트웨어 아키텍처의 로봇 서버 쪽에, RDC 계층은 로봇 클라이언트 쪽에 해당한다. RS 계층은 로봇 클라이언트로부터 제공 가능한 서비스들을 추상화하여 새로운 서비스를 조합하는 기능과 조합된 서비스를 배포하는 기능을 가지며, 로봇 디바이스 제어(RDC) 계층은 로봇 디바이스를 실제로 제어하기 위한 디바이스 드라이버가 탑재된 로봇 클라이언트를 의미한다.

그림 5는 서버-클라이언트 구조를 바탕으로 하는 하위 계층의 그림을 나타낸다. 그림 5에서와 같이 RS 계층은 로봇 서비스의 실행 및 조합의 기능을 지원하는 RSEL을

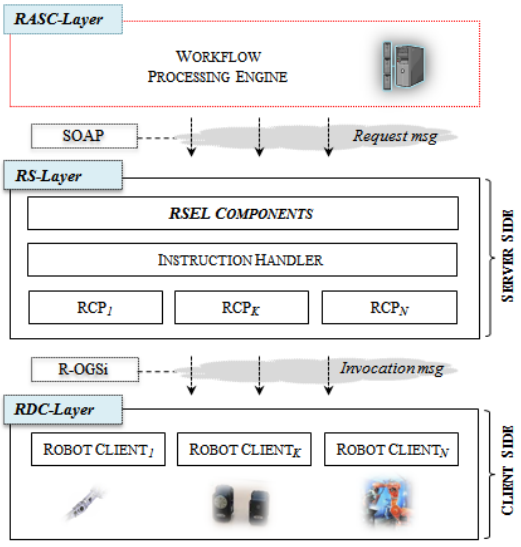

그림 5. 서버-클라이언트 계층 구조 (RS & RDC)

사용한다. RASC 계층에서는 RS 계층으로부터 RSEL을 바탕으로 배포된 로봇 응용 서비스를 호출할 수 있다. 이때 서비스는 웹 서비스 형태로 제공된다. 또한 서버 쪽에 해당하는 RS 계층은 로봇 클라이언트와의 무선 통신을 통해 다양한 이기종 로봇 시스템들을 제어할 수 있다. 이에 반해 로봇 클라이언트에 속하는 RDC 계층은 다양한 로봇 시스템을 제어하기 위한 디바이스 드라이버를 가진다. 구체적인 제어 과정은 4장 시나리오의 적용에서 그림 11과 함께 설명하도록 한다.

그림 5에서와 같이 RS 계층의 핵심 모듈은 RSEL 컴포넌트이며, 그 아래 명령어 핸들러(Instruction Handler)는 RCP(Robot Client Proxy)를 통해 로봇 클라이언트 제어를 위한 명령어를 처리하는 역할을 담당한다. 그림 6

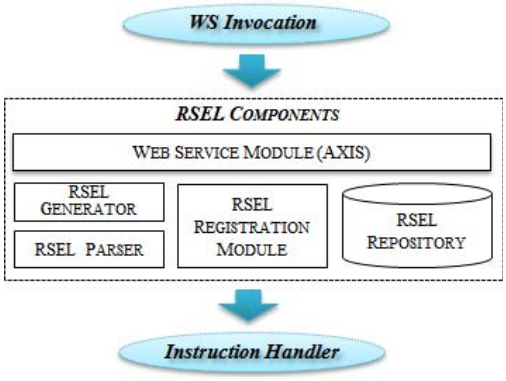

그림 6. RSEL 컴포넌트 세부 구조

은 RSEL 컴포넌트를 좀 더 상세히 표현한 것이다.

그림 6에서와 같이 RSEL의 입력은 상위 계층(RASC)로부터의 서비스 호출 메시지이며, 출력 결과는 로봇 디바이스를 제어하는 명령어 값이다. 이 명령어는 Instruction Handler를 통해 처리된다. 그림 6의 RSEL 컴포넌트는 웹 서비스 호출 인터페이스에 해당하는 AXIS 엔진, 로봇 서비스를 응용하는 기능을 담당하는 RSEL 생성기, 이를 처리하는 RSEL 파서, 그리고 생성된 RSEL 문서를 등록하는 RSEL 등록 모듈과 조합된 로봇 서비스의 재사용을 위한 RSEL 저장소로 구성된다.

4. 시나리오의 적용

본 논문에서 제안하는 계층적 로봇 소프트웨어 아키텍처의 설계 목적은 유비쿼터스 컴퓨팅 환경에서 워크플로우 기술을 통해 사용자에게 자율적인 자동화 서비스 및 각각의 역할을 담당하는 개발 계층 구조를 바탕으로 로봇 응용 서비스의 흐름을 제어 및 실행할 수 있는 환경을 제공하는 것이다.

이를 위해 본 연구에서 제안하는 아키텍처를 기반으로 “부스안내 로봇서비스”의 시나리오의 적용을 통해 로봇 서비스의 제공 가능성을 확인한다. 구체적으로는 본 연구진이 개발한 CAWL과 RSEL을 활용한다. CAWL은 상황인지 기반의 로봇 자동화 서비스를 사용자의 응용 서비스 수준에서 표현하며, RSEL은 로봇의 행동을 추상화한 로봇 서비스를 나타낸다.

4.1 시나리오의 적용 환경

본 시나리오의 적용을 위해 가정하는 로봇 서비스 제공 환경은 분산 네트워크를 기반으로 하는 컴퓨팅 환경이다. 이 환경에는 자동화 서비스를 처리할 수 있는 워크플로우 시스템과 워크플로우 시스템으로부터 실제 로봇을 제어하기 위한 정보를 전달받아 로봇을 제어할 수 있는 로봇 서버가 존재하며, 이들 시스템은 유선 네트워크를 통해 정보를 교환할 수 있다. 또한 로봇 서버와 로봇은 블루투스(Bluetooth)와 같은 무선 통신 방법을 사용할 수 있다. 그리고 사용되는 로봇의 경우 레고 로봇(LEGO Mindstorms NXT)을 사용할 수 있다. 레고 로봇의 경우 바퀴를 통한 구동 및 팔을 이용한 수직상하운동 기능이 있고 속도, 거리, 소리 등을 인식할 수 있는 센서들을 자체로 가지고 있다. 또한 사용자에게 화면으로 출력할 수 있는 디스플레이를 탑재하고 있다. 음성 출력과 같은 추가적인 기능이 필요한 경우에는 음성 출력이 가능한 디바이스를 레고 로봇에 부착하여 활용 가능하다. 그림 7은 이 같은 내용을 바탕으로 표현한 로봇 서비스 제공 환경을 나타낸다.

그림 7에서 보는 바와 같이 로봇 서비스의 제공을 위한 로봇 제어 과정은 번호 순서대로 진행된다. 위 그림에서 로봇 서버와 로봇 클라이언트에서 로봇 서비스를 배포하는 환경은 서버-클라이언트 구조를 가진다. 로봇 클라이언트는 실제로 로봇 서비스를 수행하는 로봇 디바이스이다. 서버는 로봇 서비스를 웹 서비스로서 배포하고, RSEL을 통해 로봇 서비스를 제공하는 단일 시스템이다. 워크플로우 시스템이 로봇 서버에 있는 웹 서비스를 호출하면, 로봇 서버는 로봇 클라이언트에 있는 클라이언트-사이드의 로봇 서비스를 호출한다. 기본적으로 로봇 서버는 중앙의 단일 장치로서 다른 많은 로봇 클라이언트와 웹 서비스를 가질 수 있다. 아래 그림에서 ①은 웹 서비스 호출을 나타내며, ②는 로봇 클라이언트에 있는 로봇 서비스의 호출하는 것을 의미한다.

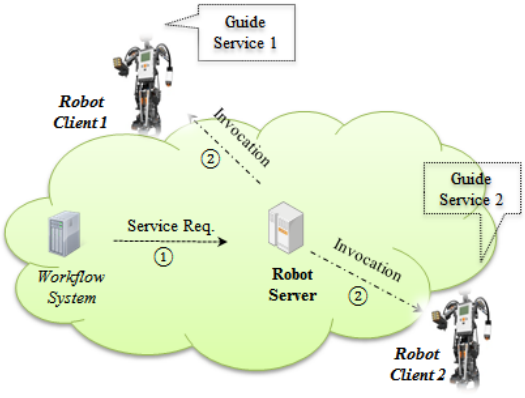

그림 7. 로봇 서비스 제공 환경

4.2 서비스 시나리오 및 워크플로우

실험을 위해 사용자가 상황 정보에 따라 로봇 자동화 서비스를 제공하기 위한 서비스 시나리오를 사용한다. 전시회에서 일반적으로 경험할 수 있는 상황을 가정하여 다음과 같은 시나리오를 작성하였다.

그림 8과 같은 서비스 시나리오를 바탕으로 로봇을 통해 제공할 수 있는 서비스의 목록은 다음과 같다.

◦ 로봇이 선택된 후 관람객에게로의 이동 서비스 (Move2User)

1. 관람객(Mr. Kim)이 행사장에 도착하여 스마트 폰 으로 가용한 로봇 서비스를 검색한 후, 검색된 서비스 중에 행사장 부스 안내 로봇 서비스를 선택함
2. 대기하고 있던 안내 로봇(R01)이 Kim에게 다가옴
3. Kim은 로봇을 통해 안내 받고 싶은 부스(B03)를 선택함
4. R01은 관람객이 선택한 부스(B03)으로 이동함
5. R01이 부스에 도착하면 B03에 대한 정보를 화면 및 음성으로 관람객에게 전달함

그림 8. BoothGuide 서비스 시나리오

- 부스 번호를 선택하면 해당 부스로의 이동 서비스 (Move2Booth)
- 부스 도착 후 부스의 전시 정보를 보여주는 서비스 (DisplayBoothInfo)
- 해당 부스 정보를 음성으로 전달하는 서비스 (VoiceService)

위 서비스 목록은 그림 9에 워크플로우 형태로 표현하였으며, 언급한 목록 중에서 DisplayBoothInfo와 VoiceService 서비스는 Flow B의 BoothInfo 서비스에 해당한다.

그림 9은 관람객이 자신의 스마트 폰으로 로봇을 검색한 후의 서비스를 워크플로우로 표현한 것이다. 그림에서와 같이 사용자의 스마트 폰을 사용하여 서비스 검색이 이루어진 후에, 해당 로봇을 검색(Search Robot)하면 워크플로우 서비스를 시작한다.

그림 10는 CAWL 기반의 워크플로우 문서를 나타낸다. 그림 9에서 타원의 점선으로 표시한 부분은 그림 9에서 표현한 각각의 서비스를 나타낸다. 그림 10에서 보는

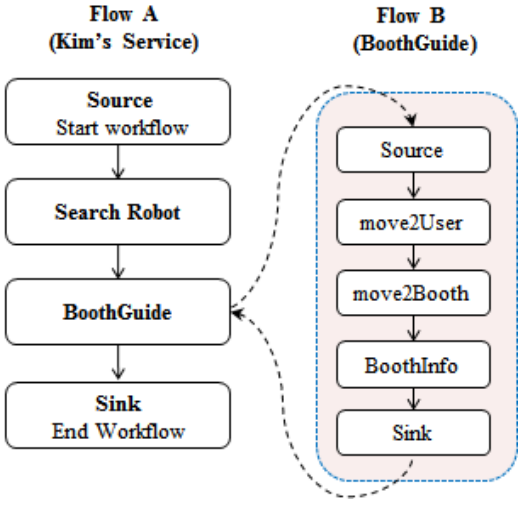

그림 9. BoothGuide 서비스 시나리오에 대한 워크플로우

```
<node name="Move2User">
  <condition expression="C1">
    <context>
      <rule contribute="positive">
        <constraint name="C1">
          <subject type="Person">Kim</subject>
          <verb>selected</verb>
          <object type="Service">BoothGuideService</object>
        </constraint>
      </rule>
    </context>
  </condition>
  <invoke serviceProvider="robotService" operation="moveToUser"/>
</node>

<node name="Move2Booth">
  <condition expression="C1">
    <context>
      <rule contribute="positive">
        <constraint name="C1">
          <subject type="Robot">R01</subject>
          <verb>isSelected</verb>
          <object type="Booth">B03</object>
        </constraint>
      </rule>
    </context>
  </condition>
  <invoke serviceProvider="robotServie" operation="moveToBooth"/>
</node>

<node name="InfoBooth">
  <condition expression="C1 and C2">
    <context>
      <rule contribute="positive">
        <constraint name="C1">
          <subject type="Robot">R01</subject>
          <verb>isLocated</verb>
          <object type="Booth">B03</object>
        </constraint>
        <constraint name="C2">
          <subject type="Person">Kim</subject>
          <verb>isLocated</verb>
          <object type="Booth">B03</object>
        </constraint>
      </rule>
    </context>
  </condition>
  <invoke serviceProvider="robotServie" operation="displayBoothInfo"/>
  <invoke serviceProvider="robotServie" operation="briefBoothInfo"/>
</node>
```

그림 10. BoothGuide 서비스의 내용을 표현한 CAWL 문서

바와 같이 CAWL 문서는 3개의 노드로 구성되어 있으며, 마지막 점선으로 표시한 BoothInfo 노드에서는 display BoothInfo와 briefBoothInfo 서비스를 동시에 호출할 수 있음을 보여준다. CAWL 문서는 각각 서비스 노드의 분기 조건은 상황 정보를 표현하는 특징을 갖는다. 그림 10에서는 이와 같은 상황 정보를 constraint 단위로 표현하였으며, 이는 {subject, verb, object}의 엔티티 집합으로 기술한다.

예를 들어 그림 10에서 세 번째 서비스 노드(Info Booth)의 경우를 다음과 같이 설명할 수 있다. 내용은 "서비스 로봇(R01)과 관람객(Mr. Kim)이 가려고 하는 부스(B03)에 도착하면, 서비스 로봇(R01)에게 자신의 스크린을 이용한 부스의 설명(displayBoothInfo) 및 음성을 통한 간단한 브리핑(briefBoothInfo) 서비스를 제공하도

록 호출한다(invoke)"는 것이다. 이 때 operation은 워크플로우 엔진에서 호출하는 웹 서비스를 표현한 것이며, 이 웹 서비스는 제안하는 소프트웨어 아키텍처의 RS-Layer에서 배포한 것을 사용한다.

4.3 로봇 서비스 응용 및 배포

본 절에서는 RS 계층에서 로봇 서비스를 조합 및 재사용할 수 있는지에 관하여 설명하도록 한다. RAS 계층에서는 사용자 수준에서의 로봇 응용 서비스를 조합하고 재사용하였다면, RS 계층에서는 RAS 계층에서 사용하게 될 로봇 서비스를 배포하는 역할을 수행하며, 이때 생성되는 로봇 서비스는 RDC 계층의 로봇 디바이스의 HW 관점에서의 서비스를 소프트웨어 관점의 로봇 서비스로 추상화한 개념이다. 이와 같은 내용을 본 논문에서는 RSEL로 표현하였으며, 내용은 다음과 같다.

RSEL의 목적은 HW 관점의 로봇 서비스를 추상화하고, 이를 소프트웨어 관점의 로봇 서비스 형태로 조합하여 상위 계층에서 사용할 웹 서비스로 제공하는 것이다. 표 1은 4.2절에서 언급했던 서비스의 목록을 실제 웹 서비스 호출할 때 필요한 매개 변수와 함께 표현한 것이다. 로봇 서비스 개발자는 표 1의 서비스들을 바탕으로 RSEL통해 부스 안내(BoothGuide) 서비스를 그림 10과 같이 작성할 수 있다. 따라서 RSEL은 복잡한 상황에 적합한 서비스를 제공하기 위해, 단순한 서비스들의 조합을 통해 좀 더 다양할 서비스를 배포할 수 있는 특징을 가진다.

그림 11은 RSEL 문서의 한 부분으로 BoothGuide 서비스를 조합한 내용을 보여 준다. 그림에서 <process> 태그는 하위 태그로 <invoke> 를 가지고 있으며, 이 <invoke> 태그는 서비스 조합의 대상이 되는 각각의 서브 서비스를 표현하는 'operation' 속성 값을 갖는다. RSEL은 로봇 서비스를 조합하는 목적을 갖는다. 따라서 그림 11에서와 같이 한 문서 내에 다수의 <invoke> 태그를 갖는다. 이 때 기술한 서브 로봇 서비스들은 순차적으로 실행된다. 따라서 BoothGuide 서비스는 moveToUSer, motoBooth, displayInformation, 그리고 briefInformation의 서브 서비스들로 조합되고, 나열된 순서대로 실행된다. 이들 서브 서비스들은 모두 RDC 계층의 로봇 클라이언트로부터 제공되며, 웹 서비스 인터페이스를 이용하여 실행된다.

표 1. BoothGuide 로봇 서비스 목록

	서비스 이름	매개 변수
1	move	-
2	moveToUser	User location
3	moveToBooth	Booth number
4	displayInformation	Booth number
5	briefInformation	Booth number

```
<RSEL layer="RS-Layer" name="BoothGuide"
  <parameter>
    <variable name="$userLocation"/>
    <variable name="$boothNumber" />
    <variable name="$processResult" />
  </parameter>

  <process>
    <!-- send data -->
    <invoke layer="RS-Layer"
            operation="moveToUser"
            inputVariable="$userLocation"
            outputVariable="$processResult"/>

    <invoke layer="RS-Layer"
            operation="moveToBooth"
            inputVariable="$boothNumber"
            outputVariable="$processResult"/>

    <invoke layer="RS-Layer"
            operation="displayInformation"
            inputVariable="$boothNumber"
            outputVariable="$processResult" />

    <invoke layer="RS-Layer"
            operation="briefInformation"
            inputVariable="$boothNumber"
            outputVariable="$processResult" />

  </process>

  <return>
    <variable name="$processResult"/>
  </return>

</RSEL>
```

그림 11. BoothGuide 서비스를 표현한 RSEL 문서

4.4 로봇 서비스의 처리 과정

본 절에서는 BoothGuide 서비스를 처리하는 과정에 대하여 살펴본다. 설명은 그림 12에서와 같이 순차적인 번호(①~⑤)로 표현하였다. 로봇 서비스를 실행하는데 클라이언트-서버의 관점에서 설명하기 위해, 그림 12에서는 RS 계층을 서버 쪽(Robot Server Side), 그리고 RDC 계층을 클라이언트 쪽(Robot Clients)으로 표현하였으며, RSEL 문서의 생성 과정은 알파벳(ⓐ~ⓒ)으로 표현하였다. 세부 내용은 다음과 같다.

① RASC 계층의 워크플로우 엔진이 로봇 서버 측의 로봇 서비스를 요청
② 사용 가능한 로봇 서비스들이 RSEL 문서로 배포됨
③ BoothGuide 서비스를 위한 RSEL 문서 내의 서브 서비스들을 Instruction Handler로 순서대로 보냄
④ ~ ⑤ Instruction Handler는 RCP를 통해 순차적으로 각각의 가이드 로봇 서비스를 실행함

세부 내용 ①의 경우 워크플로우 엔진은 그림 10의 워크플로우 문서를 그림 3의 CAWL 파서를 통해 필요한 로봇 서비스를 호출하기 위한 정보만을 추출한다. 그림 10의 타원형 점선 부분에 해당하는 각각의 operation의 값은 실제 워크플로우 엔진이 호출하는 웹 서비스이며, 이 정보는 CAWL 파서가 추출하여 그림 12의 RSEL 컴포넌트로 전달되어 진다. 이와 같이 호출된 웹 서비스는 그림 12에서 RSEL 컴포넌트의 AXIS를 통해 인식될 수 있으며, 이 때 실제 호출하려고 하는 로봇 서비스는 그림 11의 RSEL 문서에 기술한 4개의 Operation이 된다.

이 때 RCP는 로봇 클라이언트의 추상체이며, 이는 다양한 로봇 클라이언트를 수행할 수 있도록 한다. 따라서 Instruction Handler를 통한 RCP의 제어는 로봇 클라이언트를 실행시키는 것이다. 그리고 그림 12에서 ⓐ ~ ⓒ는

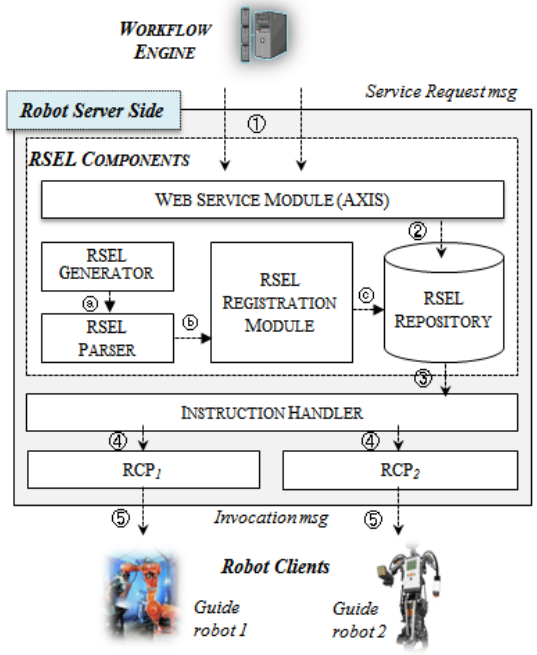

그림 12. BoothGuide 서비스 처리 과정

RSEL 문서를 생성하는 과정을 순차적으로 표현한 것이다. 우선 RSEL Generator는 RSEL 문서를 생성하고, RSEL Parser는 생성된 RSEL 문서를 파싱하여 가용 여부를 판별한다. 마지막으로 RSEL Registration Module은 조합된 로봇 서비스의 재사용을 위해 생성된 RSEL 문서를 RSEL Repository에 저장한다.

4.5 시나리오 적용 평가

본 논문에서는 제안하는 아키텍처의 실제 로봇 제어를 위한 적용 가능성을 보이기 위해 실현 가능한 시나리오를 가정하여 그 처리 과정을 살펴보았다. 우선 사용자 수준에서의 로봇 서비스를 기술하기 위한 CAWL 문서를 바탕으로 서비스 워크플로우를 보였으며, 이렇게 기술된 서비스의 내용은 그림 4의 과정을 통해 상황을 판단하고 로봇 서비스를 호출한다. 호출된 로봇 서비스는 실제 로봇의 서비스를 추상화하여 표현한 그림 11의 RSEL로 나타낼 수 있으며, 이 서비스 정보를 통해 실제 로봇을 제어할 수 있다.

5. 결 론

본 논문에서는 유비쿼터스 로봇을 이용하여 사용자에게 자율적인 자동화 서비스를 제공하기 위한 계층적 아키텍처를 제안하였다. 제안하는 로봇 소프트웨어 아키텍처는 핵심 기술로 CAWL 기반의 상황인지 워크플로우 자동화 기술과 RSEL 기반의 로봇 응용 기술을 적용하였다. 전자는 상황 정보를 바탕으로 사용자 응용 서비스 수준에서의 서비스의 흐름을 표현 및 이를 처리하는 기술이며, 후자는 로봇 디바이스 서비스를 추상화한 로봇 서비스를 조합하여 서비스의 재사용 및 확장성을 고려한 기술이다. 또한 이 두 핵심 기술을 바탕으로 한 본 논문에서 제안한 계층적 아키텍처는 RAS, RASC, RS, 그리고 RDC의 4개 계층으로 구성되어 있고, 각각의 계층은 계층 상호간 종속성을 줄이기 위해 웹 서비스 인터페이스를 통해 접근이 가능하도록 설계되었다.

실험에서는 실현 가능한 시나리오를 바탕으로 CAWL 및 RSEL을 중심으로 시나리오의 적용 가능성을 보여주었다. 그러나 실제 이와 같은 서비스를 제대로 제공하기 위한 지능형 로봇의 수준이 아직은 초기 수준에 있으므로, 매우 현실적인 시나리오의 적용 및 실현은 다소의 시간이 필요하다. 한편으로 이러한 실현 가능성은 상업화의

가능성을 열어 주기도 한다. 따라서 제안하는 로봇 소프트웨어 아키텍처는 지속적인 로봇 시스템의 개발과 함께 좀 더 실현 가능한 연구에 적용 가능할 것으로 생각한다.

참고문헌

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

[2] OPRoS, http://opros.or.kr.

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

[4] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, "A Review of Middleware for Networked Robots", International Journal of Computer Science and Network Security, Vol. l9, No. 5, pp.139-148, 2009.

[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] ERSP, http://www.evolution.com/products/ersp.

[7] 최종선, 조용윤, 최재영, "다중-워크플로우를 지원하는 상황인지 워크플로우 언어의 설계", 한국인터넷정보학회논문지, 제10권, 제6호, pp.145-157. 2009.

[8] M Wieland, O. K., D Nicklas, F Leymann, "Towards context-aware workflows", CAiSE07 Proc. of the Workshops and Doctoral Consortium, 2007.

[9] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, "Business Process execution Language for Web Services", BEA Systems, Microsoft Corp, IBM Corp, Version 1.1, 2003.

[10] Wieland M. Kaczmarczyk P., Nicklas D., "Context Integration for Smart Workflows", Proceedings of the 6th Annual IEEE International pp.239-242. 2008.

[11] Jun Li, Yingyi Bu, Shaxun Chen, Xianping Tao, Jian Lu, "FollowMe: On Research of Pluggable Infrastructure for Context-Awareness", AINA06, Vol. 1, pp.199-204, 2006.

[12] Joohyun Han, Yongyun Cho, Eunhoe Kim, Jaeyoung Choi, "A Ubiquitous Workflow Service Framework", ICCSA06: pp.30-39, 2006.

[13] 유원필, 박승환, 채희성, "u-City 로봇시스템기술 동향", 전자통신동향분석, 제24권, 제5호, pp.98-108, 2009.

[14] Hyun Kim, Young-Jo Cho, Sang-Rok Oh, "CAMUS: a middleware supporting context-aware services for network-based robots", Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on, pp.237-242, 2005.

[15] 홍충성 외 5명, "URC를 위한 상황 정보 관리 기술", 전자통신동향분석 제22권, 제2호, pp.10-19, 2007.

[16] 정연구, 조현규, "지능형 로봇의 국제 표준화 동향", 전자통신동향분석, 제22권, 제2호, pp.70-78, 2007.

[17] Tetsuo Kotoku, Makoto Mizukawa, "Robot Middleware and its Standardization in OMG", SICE-ICASE, IEEE: pp. 2028-2031, 2006.

[18] 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.

[19] Simeon Tsvetanov, "Using Some Motion Devices for Easily Workflows Illustration", International Scientific Conference Computing Science'08, 2008.

[20] 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.

[21] W. S. Harwin, R. G. Gosine, Z. Kazi, D. S. Lees, J. L. Dallaway, "A Comparison of Rehabilitation Robotics Languages and Software", Robotica, vol. 15, pp.133-151, 1997.

최 종 선

2000 숭실대학교 컴퓨터학부 (학사)
2002 숭실대학교 컴퓨터학과 (공학석사)
2010 숭실대학교 컴퓨터학과 (공학박사)
2010~현재 숭실대학교 지능형로봇연구소 전임 연구원
관심분야 : 분산 컴퓨팅, 유비쿼터스 컴퓨팅, 로봇 미들웨어

조 용 윤

1995 인천대학교 전자계산학과 (학사)
1998 숭실대학교 컴퓨터학과 (석사)
2006 숭실대학교 컴퓨터학과 (박사)
2009~현재 국립순천대학교 정보통신공학과 조교수
관심분야 : 프로그래밍 언어, 컴파일러, XML, HCI, 유비쿼터스 컴퓨팅

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 IT대학 컴퓨터학부 교수
관심분야 : 분산 컴퓨팅, 고성능컴퓨팅(HPC), 유비쿼터스 컴퓨팅