Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 201063, 10 pages
http://dx.doi.org/10.1155/2014/201063

Research Article

A Layered Middleware Architecture for Automated Robot Services

Jongsun Choi,[1] Yongseong Cho,[1] Jaeyoung Choi,[1] and Jongmyung Choi[2]

[1] *School of Computer Science and Engineering, Soongsil University, Seoul 156-743, Republic of Korea*
[2] *Department of Computer Engineering, Mokpo National University, JeollaNam-do 153-729, Republic of Korea*

Correspondence should be addressed to Jaeyoung Choi; choi@ssu.ac.kr

Received 6 January 2014; Accepted 17 April 2014; Published 26 May 2014

Academic Editor: Hoon Ko

These days, using robots has gradually been extending from the limited industrial areas in factories to service areas for the general public in everyday life. It is possible to imagine that end users easily control robots and they define new services by themselves as they wish in the near future. However, there are three issues to resolve to realize the future. Firstly, it is required to abstract robots' behaviors from primitive robot functions, and secondly, we need context awareness in order to provide users with the appropriate services. Lastly end users can define the robot services easily. In this paper, we propose three layered middleware architecture for automated robot services to resolve the problems. The lowest layer abstracts robots' behaviors in functional level, and the middle layer provides users with robot services based on the situational information. And the highest layer defines robot services by sequential order. End users can define practically a flow of robot services by using an intuitive GUI. We showed the availability and effectiveness of the layered middleware architecture with a prototype system, which presents the facile definition of robot services in application level, the control process of robot service, and the abstraction process of robot services.

1. Introduction

Due to a new paradigm of ubiquitous computing, the concept of ubiquitous robot [1] has started to appear, which is defined as a robot incorporating three components including virtual software robot or avatar, real-world mobile robot, and embedded sensor system in surroundings. Recently, studies on robot middleware and network-based robot system for providing robot services are actively conducted in many countries [2–8]. Such changes in paradigm are expanding the focus of robot from industrial robots to intelligent service robots, which provide users with appropriate services in daily life. In ubiquitous computing environment, users want to receive intelligent services from robots anytime and anywhere. Thus, the ubiquitous robot should be able to recognize its surrounding environment and provide users with smart and situation-aware services. A smart service is not a single task but a series of simple and complicated tasks, so that it is required to define a flow for the service. WS-BPEL [9] is good example for the services. Web services-based workflow technology provides means for easily integrating various robot services based on heterogeneous hardware platform, as well as for providing the low degree of coupling for configuring middleware architecture in order to reduce dependency among service modules.

A robot software platform must typically provide the environment required for performing a role such as the definition and executing of robot application services and abstraction of devices. In order to perform the role, the robot software generally has a layered architecture approach that is composed of task definition layer, task execution layer, and device abstraction layer. Device abstraction layer is for abstraction of robotic devices. Task execution layer performs the role of controlling and executing the abstracted robot services. Task definition layer defines the execution order of the abstracted robot services according to various restricted conditions.

Because robot devices are composed of complicated heterogeneous hardware, it is difficult for end users or robot service developers to define robot application services and execute them. Thus, a robot software platform has to provide a development environment to easily define various services

at the level of a robot application services. At an expert level, it is able to provide a convenient method for abstracting heterogeneous robot devices, and execution environment based on context awareness to provide the abstracted robot services according to restricted conditions, namely, situational information using contexts from surroundings.

In this paper, we propose a layered architecture to abstract heterogeneous robotic devices, define robot application services conveniently with the abstracted robot services at the end-user level, and execute them according to situational information.

The layered architecture consists of three layers: robot service layer (RS-L), robot application service control layer (RASC-L), and robot application service layer (RAS-L). RS-L generates various robot services through abstraction from heterogeneous robotic devices. RASC-L is able to control and execute the generated robot services by context awareness. And RAS-L enables robot service developers to easily define the execution order of the generated robot services at the end-user level.

As a workflow technique for automated robot services in proposed architecture, CAWL is applied to RAS-L to express the relationship of robot services with service flow, execution order, and situational information. CAWL is a context aware workflow language that can express the flow and control of robot services based on context awareness at robot application service layer. CAWL is also able to describe context-based situational information directly on a CAWL document by applying RDF-based context model. RSEL is a robot service execution language applied to RS-L for combining and reusing robot services that is abstracted from heterogeneous robotic devices. RSEL is also able to describe abstracted robot services, for control of actual robotic devices, and provides functions to combine them into different forms of robot services in a RSEL document.

2. Relevant Studies

Until recently, there are studies conducted to adapt workflow technology, an automated model for business process integration that is validated by renowned computer companies in the world such as IBM, Microsoft, BEA, and Oracle, to ubiquitous computing environment [9–13]. There are also studies on intelligent robot middleware that expanded the field of context awareness middleware studies to the field of robotics that is actively going on [14–17]. In this chapter, recent trend and associated technology on abovementioned workflow technology and studies on middleware [18–20] are examined.

2.1. Robot Software. Many countries in the world including Japan which are leading the standard of robot middleware in OMG are trying to cultivate robot industry using network-based intelligent robots. Korean government, especially, is trying to upgrade the robot technology using the top level communication infrastructure in the world. As a part of such effort, ETRI in Korea is developing intelligent service robot that actively utilizes the network environment such as RFID, UWB, and ZigBee through USN-based Ubiquitous Robotic Space task since 2005. CAMUS [14], context awareness infrastructure developed by ETRI, is a context awareness middleware for URC system. It supports the development and implementation of context-based application in order to provide appropriate service to users based on obtained context within URC environment. Currently, the government ordered the users and the developers to comply with uniform standard of OPRoS [3] robot software platform. OPRoS is a modified and enhanced robot software platform standard created by Electronics and Telecommunications Research Institute (ETRI) which started from RUPI standard and passed through CAMUS. Currently, Korea is participating in robot middleware standard activities in OMG with Japan based on such technology. OPRoS consists of OPRoS server, OPRoS communication protocol, OPRoS robot client, robot contents, and application component integrated development environment. For OPRoS robot application, OPRoS plugin or OPRoS component method is applied. Developers should create programming that creates plugin or component based on these two methods.

There is OpenRTM-aist [4, 5] ongoing in Agency of Industrial Science and Technology (AIST) in Japan, and it is selected as standard in 2006 [15]. However, low reusability is the problem with software development that supports the robot system due to its specificity. In order to solve this, RT-middleware increased the reusability of components by applying the concept of software modulation in various network-based robotic devices. In addition, there is SORA [16], that applies service-oriented architecture (SOA)-based robotics software development methodology, and this is a software structure for the robots developed in NASA in U.S.

There is also BPEL, a robot system suggested by Tsvetanov [17], which is a robot middleware with standard workflow technology applied. BPEL workflow language that describes the robot process has difficulty in expressing the context and thus, modification on BPEL to add context awareness function or another reasoning engine is needed. The biggest difference between the robot middleware mentioned earlier and the proposed robot middleware is a method to express robot services. That existing robot middleware, such as OPRoS, OpenRTM-aist, or SORA, defines the order of execution processes without expressing situational information. On the other hand, the proposed middleware makes it possible to develop robot service scenarios based on situational information on the robot application layer. We developed CAWL, a context aware workflow language, to describe such scenarios. Additionally CAWL execution engine was designed to provide a function to determine the situation by extracting context information in a surrounding environment, in order to provide the robot services based on the robot service scenario. The similar thing to the existing robot middleware is that component modules are created by abstracting the various robot devices and that the robot is thus controlled by processing those modules in order. The proposed middleware uses SOA as with other existing robot middleware.

The advantage of the proposed architecture is that the developer can describe a robot service scenario using the

context information on the application layer without professional technical knowledge about robot hardware equipment. Currently, it is difficult to express a variety of service scenarios because of performance constraints of the current robot platforms. However, the proposed robot middleware architecture will be popularly used in the near future as the hardware performance of the service robot is improved.

2.2. Workflow Model. Context4BPEL [10] expanded WS-BPEL [9] that is acknowledged as a standard workflow and suggested the platform structure to implement it. It also introduced smart workflow concept in the study that expanded this structure [11]. Modification of schema for WS-BPEL expansion is inevitable in Context4BPEL and thus, workflow process engine should be modified. FollowMe [12] is OSGi framework for ubiquitous computing environment and its purpose is to provide context workflow infrastructure that can be applied to various domains existing in ubiquitous computing environment. However, additional studies on workflow language used in FollowMe and the method of processing this language are needed. uFlow [13] is a context awareness workflow service framework based on uWDL, new ubiquitous workflow language, that solves the problem with existing web service based workflow languages not being able to describe the context under service transition condition.

uWDL is based on structural context model to specify the context on constraints of transition state of the workflow in order for the workflow to recognize the context on surroundings and to be able to transfer the state. Through uWDL, users' service demands are processed and implemented according to contexts in the scenario. CAWL is a workflow language for context awareness with modified overall uWDL schema structure to improve flaws of uDWL. In CAWL, context information is statically described in scenario document and the problems of not being able to express different workflow service are fixed. Through this, not only services using various u-systems existing in ubiquitous computing environment but also intelligent robot services are expressed. In this study, such improved CAWL is utilized as infratechnology to express robot application service in user level.

2.3. Robot Service Application Method. Generally, robot middleware requires hierarchal structure middleware to control robot system. Robotic device is the lowest layer of robot middleware and there are other robot application layers with little differences in their names. There have been numbers of studies carried out on robot service application methods using XML since 10 years ago especially for the design of robot application layers. Details are as follows.

RoboML [18] is the typical example for XML-based root service application and it is based on KQML [19], whose purpose is an agent communication. Also, RoboML provides methods for robot service application as well as the base for the users to utilize robot services through internet by using Embedded Agent, Interface Agent, and Proxy Agent. However, there are difficulties in providing service combination function to upper level robot service for supporting various robot services. CURL (Cambridge University Robot Language) [20] made by Cambridge university is a programming language for robot system that focuses on human-oriented programming. Grammar used in CURL is similar to human language form that makes it easy for CURL developers to understand. CURL is used as a tool for controlling the support system. It is not suitable for the means of developing robot application services. As described above, there are a number of efforts for providing various services under ubiquitous computing environment carried out in different forms including workflow technology for service automation, robot middleware technology for providing intelligent robot service, and XML-based application technology for the robot system. In this paper, abovementioned technologies serve as the main theme. Combination of robot service through the abstraction of CAWL, reflecting workflow technology, and robotic device services can be applied to software architecture which suggests RSEL that provides expandability to more complex and upper services.

3. Proposed Layered Architecture

In order to provide context awareness based robot service to the users in ubiquitous computing environment, computing environment that can express and process different services is needed. Suggested layered architecture has two main focuses, namely, CAWL and RSEL mentioned in relevant studies. The layered architecture consists of CAWL and RSEL as key components and the rest of them are modules or engines to process the key components. As is shown in Figure 1, suggested robot software architecture consists of 3 layers such as RAS-Layer, RASC-Layer, and RS-Layer. The outline of suggested architecture is as below diagram.

3.1. Robot Application Service Layer (RAS-L). RAS-L is the highest layer in suggested layered architecture and it defines robot services at user level. CAWL is a context aware workflow language that can describe user level application services based on the context. In this paper, CAWL is used to describe robot services. Below diagram is the concept map for CAWL.

In Figure 2, CAWL enables a service developer to select suitable services based on various contexts on such as profile for users and devices, location and time. It can also control workflow service flow and express various contexts from surroundings. Service developer should be sure to write multiple CAWL-based documents in order to support various workflow services for multiple users, and the context is described based on context model. The developer can write workflow scenario, namely, a robot service, or she/he can conveniently organize workflow-based robot services through intuitive UI using development tool for robot application services. The development tool is described in detail in Section 4.4.

3.2. Control of Robot Application Service (RASC-Layer). RASC-Layer is the middle layer in suggested software architecture and it controls workflow service flow defined in the upper layer (RAS-L) and invokes each robot service for

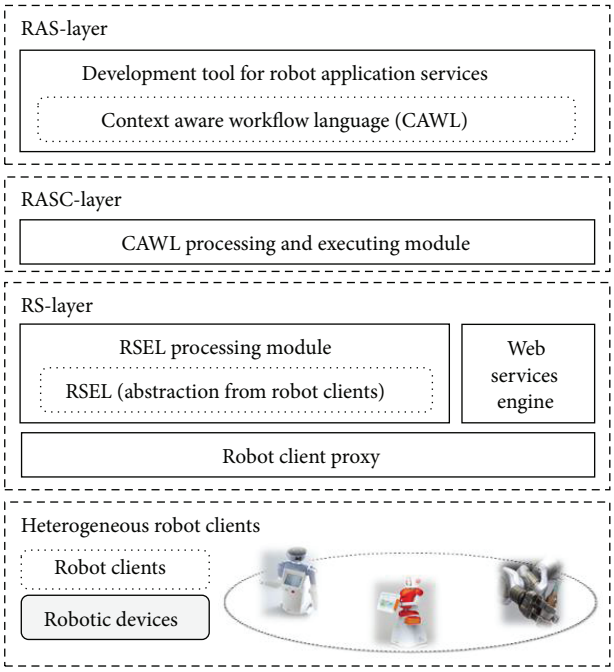


Figure 1: Outline of suggested layered architecture.

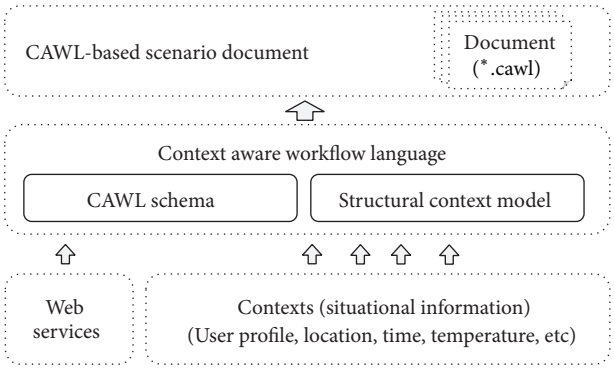


Figure 2: Conceptual architecture of CAWL.

implementation. As is shown in Figure 3, RASC-layer is based on RDF-based structural context model to express the context and the robot service is invoked based on this. Also, it uses CAWL document and defines robot application services at user level, as an input value. Based on CAWL document, the meaning of robot service flow according to the context is analyzed and appropriate robot services are provided to users accordingly.

CAWL Parser is used for parsing scenario document created with CAWL. Through this, different variables or workflow associated information needed for implementation of robot services is extracted from scenario document. Variables are managed using symbol table and the workflow information is maintained in AST (Abstract Syntax Tree) form. Saved variables are used as information to execute workflow service and AST information for workflow services can be used for reusing. In order to provide appropriate services to users according to situational information, it is needed to compare *situational information*, namely, *contexts*, from various sensors with the contexts described in scenario document. In Figure 3, context comparator compares the contexts that can be obtained from the sensor with the situational information described in CAWL document based on structural context model. Details of structural context model can be verified in uFlow developed by our research team in 2006.

Figure 4 shows the process of comparing the context in subject, verb, and object obtained from sensors and went through structural context model to context described in CAWL scenario document by application service developer. As is shown in Figure 4, the context shown in user context and scenario is represented as RDF-based triplet composed of subject, verb, and object. Thus, context information obtained from the sensor network can be objectified as sets of entities that consist of subject, verb, and object.

In Figure 4, inputs for context comparison module are scenario information described in CAWL document at the left side of Figure 4 and objectified context information at the lower side of *context comparator* in Figure 4. For example, the scenario describes the process of determining "Is a person that corresponds to `Person` located at `OfficeRoomNum`?" Information on `Person` and `OfficeRoomNum` is not static information whose value is determined in the document, but it is the value obtained from the sensor that is entered after determining the information on the person and the office. Such information creates a document instance AST through CAWL parser of Figure 3, and this information becomes input value for context comparison module. High level context information at the right side of Figure 4 is objectified information of contexts on actual person and office number by structural context model. Thus, context comparison module can determine the surroundings of the user through comparison between these input values.

3.3. Robot Services and Robotic Device Control. In Figure 3, RS-Layer *(Robot Services-Layer)* corresponds to the robot server side of suggested layered architecture, and the lower side of RS-Layer corresponds to the robot client side. RS-layer provides a new service by abstracting available services in robot clients and distributes combined services. It also includes device drivers, namely, *robot client proxy*, to control heterogeneous robotic devices. RS-layer, as is shown in Figure 3, has RSEL that supports the implementation and combination of robot services. RASC-layer is able to invoke the robot service that is generated by using RSEL in RS-Layer. Detailed processing procedure will be explained with Figure 8 in Section 4.5.

4. Application of Scenario

The purpose of designing the suggested layered architecture is to provide the environment that enables the definition of robot application services with intuitive GUI, the control of robot application services, and the abstraction of heterogeneous robotic devices. In order to actualize this, we

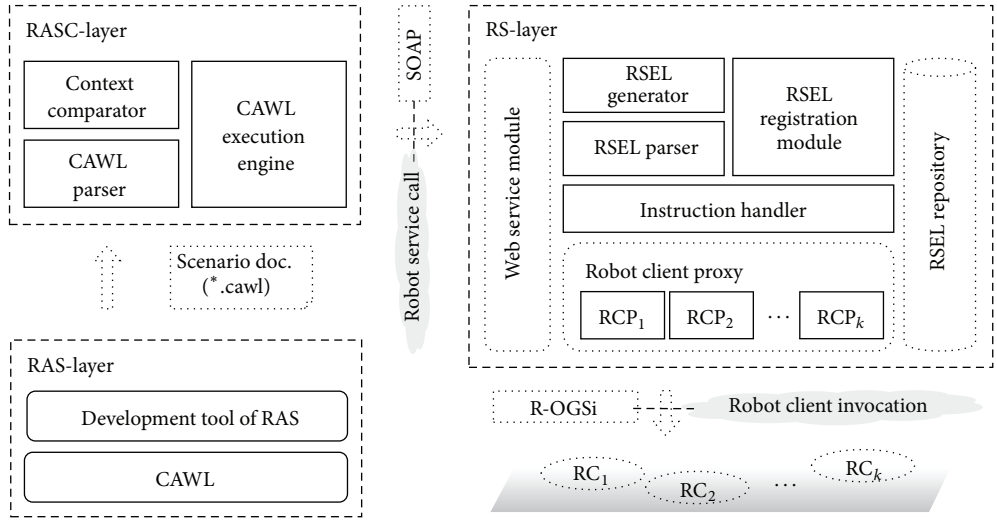


Figure 3: Layered system architecture for the definition of robot application services, execution, and abstraction of robot services.

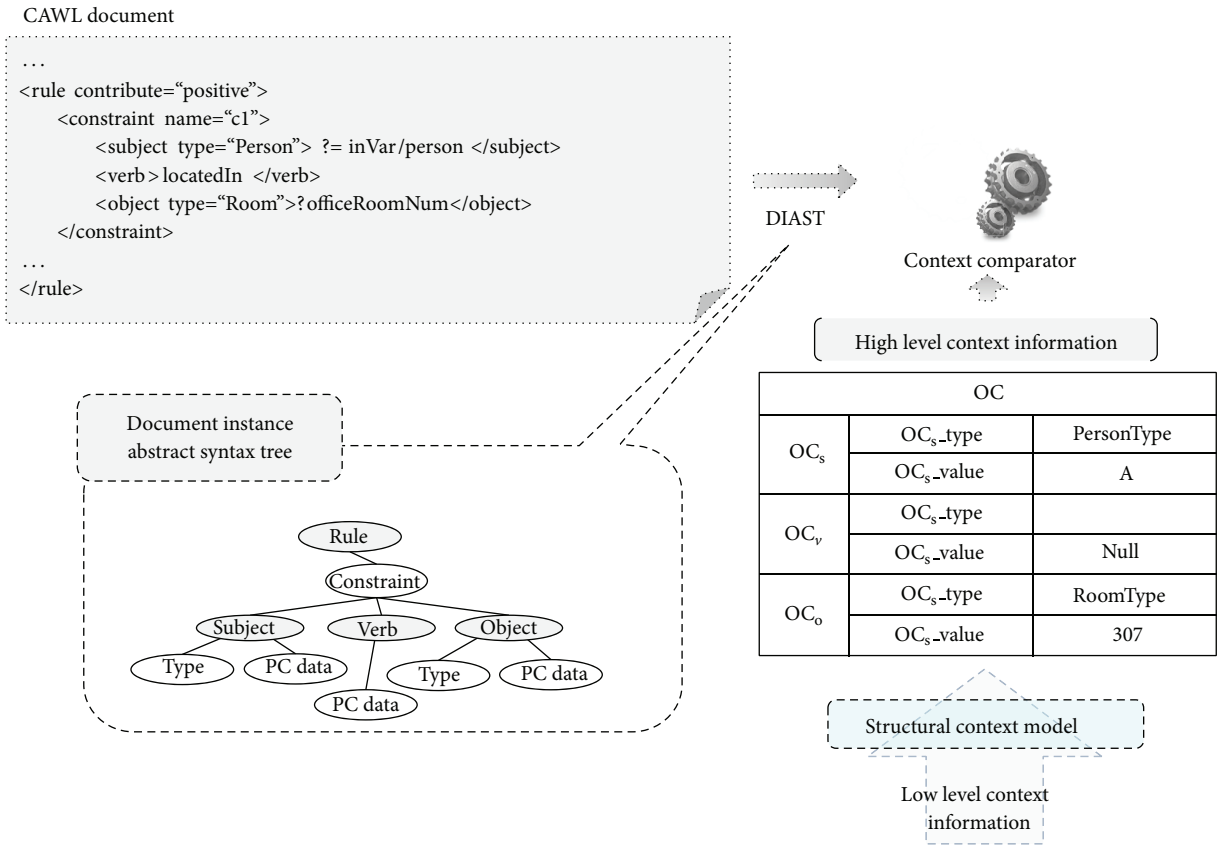


Figure 4: Extraction of context and comparison process.

verify the feasibility of providing robot service through the application of "*BoothGuide*" service scenario based on layered architecture suggested in this study.

4.1. Environment for Applying the Scenario. Robot service environment hypothesized for the application of this scenario is distributed network-based computing environment. The environment includes workflow system and robot server that can control the robot client by receiving the information from workflow system. These systems can exchange information on wired network. Wireless communication such as Bluetooth for data exchange between robot server and robot client can be applied.

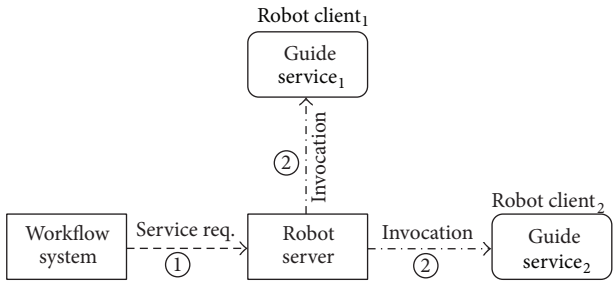


Figure 5: Overview of execution process for *BoothGuide* service.

As is shown in Figure 5, the processing procedure for controlling the robot service is carried out in consecutive order of the numbers. The environment for distributing robot service at robot server and robot client is based on server-client structure as in Figure 5. Robot client is a robotic device that actually carries out the robot service. Robot server distributes the robot service abstracted by RSEL as web services. When workflow system invokes web services generated by robot server, the robot server invokes the client-side robot service. In Figure 5, ① is to invoke web services distributed by the robot server, and ② shows the invocation for control of robot clients, namely, robotic devices.

4.2. Service Scenario and Workflow. For the experiment, the user uses the service scenario for producing robot automated services according to situational information. We have a following "*BoothGuide*" service scenario for robot control with web services. Below scenario is for the situation that can generally occur during the exhibition.

Tourist (John) arrives to the event hall and searches for the robot service available using his smartphone. Then he selects the robot service for event hall booth guide. Guide robot (R01) on standby approaches John. John selects the booth (B03) by assistance of the robot. R01 moves to the booth (B03) selected by the tourist. When R01 reaches the booth, it provides the information on B03 to the tourist through the screen and the voice.

Depending on *BoothGuide* service, the list of services that can be provided through the robot is as follows.

(i) *move2User*: moving service to tourists after selecting the robots,

(ii) *move2Booth*: moving service to corresponding booths after selecting the booth number,

(iii) *displayBoothInfo*: service that shows exhibition information of the booth after arriving to the booth,

(iv) *voiceService*: service that transfers corresponding booth information by voice transmission.

The list of robot services is shown as workflow in Figure 6. *BoothInfo* service in Flow B refers to both *displayBoothInfo* and *voiceService* of the service list. Figure 6 shows workflow services after the tourist searches for the robot using his smart phone. As is shown in the workflow, when the user searches for the service using his smart phone and selects *search Robot*, workflow service starts.

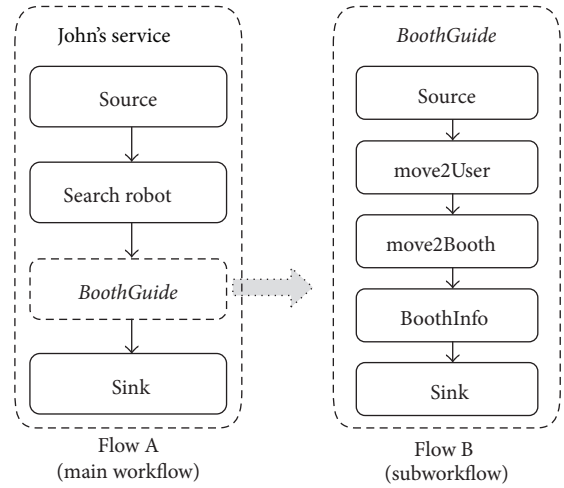


Figure 6: Workflow on *BoothGuide* service scenario.

Algorithm 1 represents CAWL based workflow document. CAWL document, in Algorithm 1, consists of 3 nodes that refer to *Movet2User*, *Move2Booth*, and *InfoBooth*. *BoothInfo* node means that both *displayBoothInfo* and *briefBoothInfo* services can be invoked at the same time. Branch condition in each service node of CAWL document represents situational information with contexts. Such contexts are shown in *constraint* unit and it is described in an entity set composed of *subject*, *verb*, and *object*.

For example, third service node (*InfoBooth*) in Algorithm 1 can be explained as follows. The content is, "when the service robot (R01) and the tourist (John) arrive to booth (B03), their destination, service robot (R01) is invoked to provide explanation (*displayBoothInfo*) of the booth using its screen and simple briefing (*briefBoothInfo*) through audio." The *operation* indicates web services invoked by workflow engine.

4.3. Application and Distribution of Robot Service. This section provides an explanation on whether to combine and reuse the robot services at RS-layer. If robot application services are combined and reused at RAS-layer, RS-layer distributes the robot service that is used in RAS-layer. The robot service generated in this layer means that it is abstracted from heterogeneous robotic devices. These robot services are represented as RSEL in this paper, and the contents are as follows.

RSEL is for combining and reusing robot services that is abstracted from heterogeneous robotic devices. The other purpose of RSEL is to provide the abstracted robot services as web services that are going to be used in upper layers such as RASC-Layer and RAS-Layer. Table 1 shows the list of services mentioned in Section 4.2 along with the parameters needed for invoking actual web services. Robot service developer can create *BoothGuide* service as Algorithm 1 by using abstracted robot services based on the services in Table 1. Thus, RSEL has a feature for providing various services through combination of a lot of services.

```
CAWL Document                                   //Combined with robot services

<node name="Move2User">
  <condition expression="C1">
    <context>
      <rule contribute="positive">
        <constraint name="C1">
          <subject type="Person">John</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="robotService" 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>
        <constaint name="C2">
          <subject type="Person">John</subject>
          <verb>isLocated</verb>
          <object type="Booth">B03</object>
        </constraint>
      </rule>
    </context>
  </condition>
  <invoke serviceProvider="robotService" operation="displayBoothInfo"/>
  <invoke serviceProvider="robotService" operation="briefBoothInfo"/>
</node>
```

Algorithm 1: Part of CAWL Document for *BoothGuide* services.

Algorithm 2 is the part of RSEL document that shows combined contents of *BoothGuide* service. <*process*> tag has <*invoke*> as sub-tag. <*invoke*> tag has "*operation*" attribute value to represent each subservice. Because RSEL should be sure to combine robot services, <*process*> tag has multiple <*invoke*> tags within one document as is shown in Algorithm 2. Such sub-robot services described in <*invoke*> tags are executed in consecutive order. In Algorithm 2, *BoothGuide* service consists of subservices including *moveToUSer*, *moveToBooth*, *displayInformation,* and *briefInformation*, and they

```
RSEL Document                                   //Combined with robot services
 <?xml version="1.0" encoding="UTF-8"?>
<RSEL layer="robotService" name="Boothguide"
   ...
   <parameter>
            <variablename="$userLocation"/>
            <variablename="$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="$processResult"/>
      <invoke layer="robotService" operation="briefInformation"
            inputVariable="$boothNumber" outputVariable="$processResult"/>
   </process>
   <return>
        <variable name="$processResult"/>
   </return>
</RSEL>
```

Algorithm 2: Part of RSEL Document.

Table 1: *BoothGuide* list of robot service.

Number	Service name	Parameter
1	*move*	—
2	*moveToUser*	*User location*
3	*moveToBooth*	*Booth number*
4	*displayInformation*	*Booth number*
5	*briefInformation*	*Booth number*

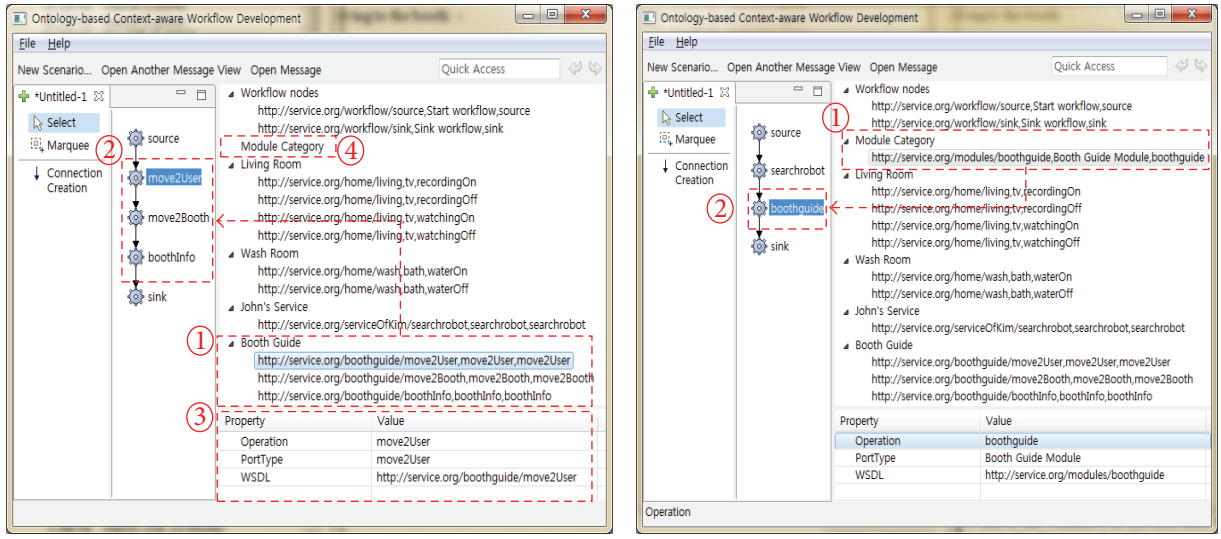
are executed with the list arranged. All these subservices are provided from RS-Layer and they are executed by using web services interface.

4.4. Definition of Robot Application Services. In this Section, we are going to present the simple process of definition of robot application services in the level of end user. Based on the scenario which was mentioned in Section 4.1, end user or robot application service developer can easily define the new robot application services by using development tool for robot application services which is presented on Figure 1 and also can develop new service by employing the existing definition of robot application services.

Figure 7(a) shows how to easily define the *BoothGuide* robot application services by using development tool, and the process is as follows. First, service developer inputs web services' URL which has the definition of robot service or WSDL document in development tool. As you can find in Figure 7(a)-①, development tool automatically generates service Nodes to develop robot application services by using entered URL or WSDL document. After producing Nodes to develop robot application services, end user or robot application service developer can develop robot application services by using drag-and-drop as it is shown in Figure 7(a)-②. Meanwhile all the necessary information for the execution of robot application services shall be entered in Property Window, Figure 7(a)-③. Figure 7(a)-④ is Module Category which installs *BoothGuide* service and its detailed chapter and verse are in Figure 7(b).

Figure 7(b) shows how to define John's Service, one of new robot application service by reusing the *BoothGuide* service. The process of defining John's Service is identical with the above-mentioned Figure 7(a) except Module category. Formerly developed robot application service will be registered on Module Category. You can find the registered services on Module Category which is in Figure 7(b)-① and *BoothGuide* service is registered on Figure 7(b). All the robot application services registered on Module category can be used as service nodes; service developer can easily reuse them by using drag-and-drop as shown in Figure 7(b)-②.

4.5. Execution Procedure of Robot Services in RS-Layer. In this section, the processing procedure of *BoothGuide* service is examined. The processing procedure is shown with consecutive numbers (① ~ ⑤) as Figure 8. In order to explain the robot service with client-server model, RS-layeris represented



(a) Definition of BoothGuide service

(b) John's service reusing the BoothGuide.

Figure 7: Definition of robot application services in RAS-Layer.

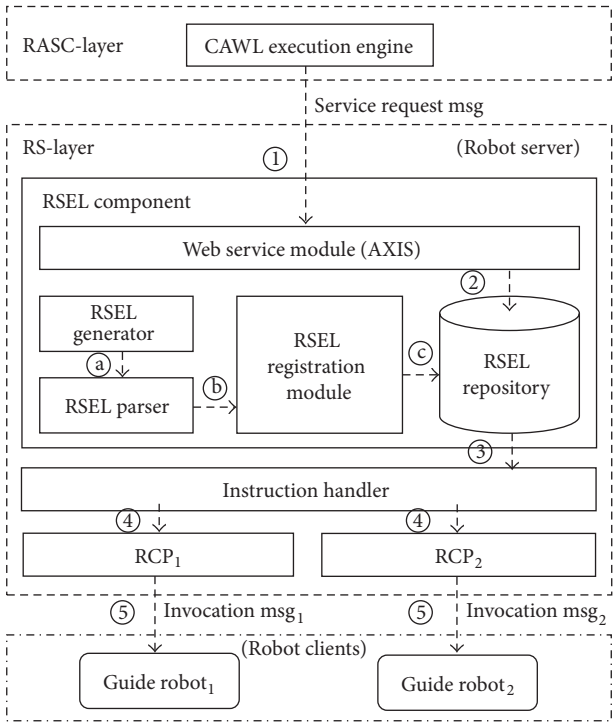


Figure 8: Processing procedure for the execution of *BoothGuide* services.

as *robot server* side and guide robots are represented as *robot clients* in Figure 8. Also, RSEL document generation process is shown in alphabets (ⓐ~ ⓒ). Details are as follows.

① Workflow engine in RASC-layer requests robot services for executing web services generated by the robot server.

② Depending on RSEL, available robot services are distributed.

③ Subservices described in RSEL document for executing *BoothGuide* service are sent to *Instruction Handler* in order.

④ ~ ⑤ *Instruction Handler* invokes each guide robot service through RCP (Robot Client Proxy) in consecutive order.

In Figure 3, CAWL parser within workflow engine extracts the information from CAWL document, in Algorithm 1, for invoking required robot services. Each operation value, such as *moveToUser*, *moveToBooth*, *displayBoothInfo*, and *briefBoothInfo*, in Algorithm 1 is web services that workflow engine invokes. This information is extracted by CAWL parser and sent to *RSEL component* in Figure 8. Such web services are recognized through web service module (AXIS) of RSEL component and the robot services that are invoked correspond to 4 operations described in RSEL document in Algorithm 2.

In Figure 8, RCP is a cone for robot client and it enables various robot clients to be executed. Thus, control of RCP by *Instruction Handler* indicates the execution of a robot client. ⓐ ~ ⓒ in Figure 8 is a representation of RSEL document generation in consecutive order. RSEL Generator creates RSEL document, and RSEL Parser parses RSEL documents generated and then examines their feasibility. Lastly, RSEL Registration Module registers the generated RSEL document to RSEL Repository for reusing combined robot service.

5. Conclusion

In this study, three layered middleware architecture for automated robot services is suggested. The lowest layer abstracts robots' behaviors in functional level, and the middle layer provides users with robot services based on the situational

information. And the highest layer defines robot services by sequential order. End users can define practically a flow of robot services by using an intuitive GUI. In suggested layered architecture, CAWL-based process automated technology and RSEL-based robot application technology are applied as key technologies. The former is a technology that describes and controls a flow of robot services based on situational information. The latter is a technology that is for reusability and expandability of the abstracted robot services by combining them. Also, the suggested layered architecture consists of three layers such as RAS-Layer, RASC-Layer, and RS-Layer. Each layer is designed to allow the access through web service interface in order to reduce mutual dependency between layers.

In the experiment conducted, possibility for applying plausible scenario focused on CAWL and RSEL is verified. However, the level of intelligent robot for providing such services is still at its early stage and thus, realistic application and actualization of this scenario may take some time. On the other hand, such possibility for actualization opens up the possibility of commercialization of the robot. Therefore, suggested robot software architecture can be applied to more plausible studies along with continuous development of the robot system.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013R1A1A2012118).

References

[1] J.-H. Kim, K.-H. Lee, Y.-D. Kim et al., "Ubiquitous robot: a new paradigm for integrated services," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy, April 2007.

[2] I. Zubrycki and G. Granosik, "Using integrated vision systems: three gears and leap motion, to control a 3-finger dexterous gripper," in *Recent Advances in Automation, Robotics and Measuring Techniques Advances in Intelligent Systems and Computing*, vol. 267, pp. 553–564, 2014.

[3] Y.-H. Choi, J.-W. Lee, S.-J. Yun, J.-H. Suh, S.-H. Hong, and J.-D. Lee, "FA system integration using robotic intelligent components," in *Proceedings of the 9th Asian Control Conference (ASCC '13)*, Istanbul, Turkey, June 2013.

[4] N. Ando, T. Suehiro, and T. Kotoku, "A software platform for component based RT-system development: openRTM-aist," in *Simulation, Modeling, and Programming for Autonomous Robots*, pp. 87–98, 2008.

[5] G. Biggs, N. Ando, and T. Kotoku, "Native robot software framework inter-operation," in *Simulation, Modeling, and Programming for Autonomous Robots*, vol. 6472 of *Lecture Notes in Computer Science*, pp. 180–191, 2010.

[6] N. Mohamed, J. Al-Jaroodi, I. Jawhar, and S. Lazarova-Molnar, "A service-oriented middleware for building collaborative UAVs," *Journal of Intelligent & Robotic Systems*, vol. 74, pp. 309–321, 2014.

[7] W. Decré, H. Bruyninckx, and J. De Schutter, "An optimization-based estimation and adaptive control approach for human-robot cooperation," *Experimental Robotics Springer Tracts in Advanced Robotics*, vol. 79, pp. 1–15, 2014.

[8] T. -H. Chou, "The service design of intelligent robot (iRobot) for entertainment," in *Proceedings of the IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW '13)*, Kyoto, Japan, July 2013.

[9] T. Andrews, F. Curbera, H. Dholakia et al., "Satish Thatte (Editor)," 2003, http://www.oasis-open.org .

[10] M. Wieland, O. K. D. Nicklas, and F. Leymann, "Towards context-aware workflows," in *Proceedings of the Workshops and Doctoral Consortium (CAiSE '07)*, 2007.

[11] M. Wieland, P. Kaczmarczyk, and D. Nicklas, "Context integration for smart workflows," in *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '08)*, pp. 239–242, Hong Kong, China, March 2008.

[12] S. Chen, J. Ge, X. Tao, and J. Lu, "A transaction model for context-aware applications," in *Advances in Grid and Pervasive Computing*, vol. 4459 of *Lecture Notes in Computer Science*, pp. 252–262, 2007.

[13] J. Han, Y. Cho, E. Kim, and J. Choi, "A ubiquitous workflow service," in *Computational Science and Its Applications—ICCSA 2006*, pp. 30–39, 2006.

[14] H. Kim, Y.-J. Cho, and S.-R. Oh, "CAMUS: a middleware supporting context-aware services for network-based robots," in *Proceedings of the IEEE Workshop on Advanced Robotics and its Social Impacts*, pp. 237–242, June 2005.

[15] T. Kotoku and M. Mizukawa, "Robot middleware and its standardization in OMG—report on OMG technical meetings in St. Louis and Boston," in *Proceedings of the SICE-ICASE International Joint Conference*, pp. 2028–2031, Busan, South Korea, October 2006.

[16] L. Fluckiger, V. To, and H. Utz, "Service oriented robotic architecture supporting a lunar analog test," in *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS '08)*, 2008.

[17] S. Tsvetanov, "Using some motion devices for easily workflows illustration," in *Proceedings of the International Scientific Conference Computing Science*, 2008.

[18] M. Makatchev and S. K. Tso, "Human-robot interface using agents communicating in an XML-based markup language," in *Proceedings of the 9th IEEE International Workshop on Robot and Human Interactive Communication RO-MAN2000*, pp. 270–275, September 2000.

[19] S. Li, Mieczyslaw, and M. Kokar, "Agent communication language," in *Flexible Adaptation in Cognitive Radios*, Analog Circuits and Signal Processing, pp. 37–44, 2013.

[20] M. Hillman, "2 rehabilitation robotics from past to present—a historical perspective," in *Advances in Rehabilitation Robotics*, vol. 306 of *Lecture Notes in Control and Information Science*, pp. 25–44, 2004.