Int J Soc Robot (2009) 1: 19–28
DOI 10.1007/s12369-008-0005-z

ORIGINAL PAPER

Multi-Layer Architecture of Ubiquitous Robot System for Integrated Services

Jong-Hwan Kim · In-Bae Jeong · In-Won Park · Kang-Hee Lee

Accepted: 28 October 2008 / Published online: 14 November 2008

Abstract Ubiquitous robot consists of various software and hardware platforms with various functions with different interfaces. In practical sense, it is impossible for software agents to hold all information related to other sensors or mobile robots. This makes it difficult to develop software agents and, at the same time, implement both modularity and scalability for the entire system. This paper proposes a novel multi-layer architecture to address the problems of interoperability between different hardware and software platforms in an ubiquitous environment offering services to the user regardless of space or time constraints. The proposed architecture is composed of five layers, which are classified according to device and environment independency, for modularity, scalability and interoperability between different hardware and software platforms. To show the general feasibility of the proposed architecture, this paper presents case studies in a simplified environment by computer simulation and experiments.

J.-H. Kim (✉) · I.-B. Jeong · I.-W. Park
Robot Intelligence Technology Lab, Department of Electrical Engineering and Computer Science, KAIST, 373-1, Guseong-Dong, Yuseong-Gu, Daejeon, 305-701, Republic of Korea
e-mail: johkim@rit.kaist.ac.kr

I.-B. Jeong
e-mail: ibjeong@rit.kaist.ac.kr

I.-W. Park
e-mail: iwpark@rit.kaist.ac.kr

K.-H. Lee
Fundamental Technology Team, Mechatronics & Manufacturing Technology Center, Corporate Technology Operations, Samsung Electronics Co., Ltd., 416, Maetan-3Dong, Yeongtong-Gu, Suwon-City, Gyeonggi-Do, Republic of Korea
e-mail: kanghee76.lee@samsung.com

Keywords Ubiquitous robot (Ubibot) · Software robot (Sobot) · Embedded robot (Embot) · Mobile robot (Mobot) · Middleware · Multi-layer architecture of ubiquitous robot

1 Introduction

Ubiquitous computing, coined by Mark Weiser, addressed the evolution of computer technology in terms of the relationship between the technology and humans [1, 2]. Emergence of mobile phone, wearable computer and ubiquitous computing predicts human beings will live in a ubiquitous world in which all devices are fully networked. The existence of ubiquitous space resulting from the developments in computer and network technology will provide motivations to offer desired services by any IT device at any place and time through user interactions and seamless applications. This shift has hastened the ubiquitous revolution, which has further manifested itself in the new interdisciplinary research area, ubiquitous robotics. It initiates the third generation of robotics following the first generation of the industrial robot and the second generation of the personal robot.

In the personal robot era based on individual robot systems, the previous research was mainly focused on three categories: artificial intelligence (AI), human-robot interaction (HRI) and dexterous mobility. Increasing advances in the field of AI robotics delivered a variety of applications for HRI and collaboration [3]. Through HRI, robot develops into a partner that enhances physical ability of human beings and possesses social abilities such as emotions [4]. In addition, network controlled intelligent mobile robot system is introduced to solve Internet latency and local intelligence of robots [5]. In this regard, studying AI, HRI and mobility

correspond to enhancing cognitive intelligence, social intelligence and behavioral intelligence of robot, respectively.

Robotic system supporting seamless services is introduced, which can be interoperable with the sensors and devices in current service environment automatically [6]. However, mobile robots have a number of constraints that limit their ability to become fully ubiquitous in physical space. Thus, the concept of ubiquitous robot (the third generation of robotics) is established in order to support seamless services even when the environment changes and to possess different characteristics of software robot, embedded robot and mobile robot [7, 8]. Software robot possesses cognitive and social intelligences. Embedded robot holds ambient intelligence, which allows to integrate various sensor information and to understand the environmental situation. Lastly, mobile robot has behavioral intelligence to offer physical services for general users and specific functions within ubiquitous space.

The ubiquitous robot environment, where all devices and objects are interconnected through a network, requires embedded robots to be a kind of a sensor system using camera, microphone, radio-frequency identification (RFID) tags, etc. The ubiquitous robot environment is implemented in the surroundings, where software robots process information and make decision, and mobile robots execute physical services. Software robots do not hold any knowledge related to physical devices in the ubiquitous space. Instead, through middleware, they make a scenario to offer the right service and determine whether the resulting service was a success or a failure. Note that receiving a proper service from the right functional software robot is equivalent to installing a right program in order to conduct proper operations when the user is using computer.

There have been numerous studies that can be directly applied to the ubiquitous robotics. Wireless sensor networks are introduced to sense the environmental situation with extendable structures [9, 10]. Viewing sensor networks as a distributed database system and applying related techniques to sensor networks is suggested [11]. The Object Management Group, Inc. (OMG) defined the Common Object Request Broker Architecture (CORBA), which uses an interface definition language to provide interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems [12]. The benefits of CORBA include language and operating system independency, technology and data transfer autonomy. The OMG also presented OMG embedded intelligence specifications, which provide networking infrastructures for a distributed community of devices and software components [13]. In addition, context synthesizing with ontologies is suggested for context-aware agents [14]. These researches provide a novel method which connects multiple devices and distributed environments, but there is a lack of researches related to actual applications and services toward user in a distributed environment with multiple devices.

This paper proposes a novel multi-layer architecture of ubiquitous robot system to address the problems of interoperability between different hardware and software platforms in a ubiquitous robot environment. The proposed architecture is composed of five layers, which are classified according to device and environment independency, for modularity, scalability and interoperability between different hardware and software platforms. Consequently, this system contributes to the development of ubiquitous robotics. We will explore the most important functions of the ubiquitous robot; perception, intelligence and action; where these functions will be both individualized and realized using embedded robot, software robot and mobile robot, respectively. The feasibility of implementing the multi-layer architecture of ubiquitous robot system is verified through both simulations and experiments in a networked environment.

This paper is organized as follows: Section 2 describes brief review on ubiquitous robot including its essential components of them. Section 3 proposes the multi-layer architecture of ubiquitous robot and its overall structure. Section 4 describes both simulation and experiment results with implementations of ubiquitous robots, and concluding remarks follow in Sect. 5.

2 Preliminaries on Ubiquitous Robot

Ubiquitous space (U-space) is an environment where ubiquitous computing is realized and every device is fully networked. Ubiquitous robot (Ubibot) is developed based on ubiquitous computing technology and works within U-space. Ubibot provides various services at any place and anytime through wired or wireless broadband network in real-time [15].

The primary advantage of the Ubibot is that it permits abstraction of intelligence from the real world by decoupling information through perception and action capabilities. While personal robot is mainly triggered by external sources, Ubibots will have the ability to understand what the user needs, wants or prefers even without direct commands and to supply continuous and seamless service. Integrated services and solutions include ubiquitous home services for security and safety, location based services like GIS, health services in tele-medicine, ubiquitous learning systems, ubiquitous commerce services, etc. [16].

Ubibot consists of embedded robots (Embots), software robots (Sobots) and mobile robots (Mobots) in various forms which specialize in perception, intelligence and action, respectively [17]. Sobot is a software system whereas Embot

and Mobot belong to hardware systems. The core intelligence of Ubibot is constituted by Sobots. Embots are embedded in U-space and ensure that Sobots possess context aware perceptive capabilities. Lastly, Mobots act upon the service requests in the physical domain. Each component of Ubibot possesses specific intelligence and roles, where this information is shared and exchanged through the network.

Simply, Ubibot generalizes to a networked cooperative robot system. Note that developing middleware is essential, which acts as an impartial link between applications, allows for transparent connectivity, and maintains secure interactions between heterogeneous protocols and various networks. General properties and technical aspects of Ubibot including Sobot, Embot, Mobot and middleware are described in the following.

2.1 Software Robot: Sobot

Sobot has all the elements of being a robot with cognitive and social intelligences for self-learning, context-awareness and calm and seamless interaction; but only virtually. Therefore, Sobot can move easily within a network and connect to other systems without time or geographical limitations. Sobot also possesses genetic intelligence, which is encoded as computerized deoxyribonucleic acid (DNA) codes to contain genetic information such as personality [18].

The core intelligence of Ubibot is constituted by Sobot, which is able to recognize the prevalent situation, make decisions on the course of action and implement them without directly consulting the user each time [16, 19]. Due to the diversity and complexity of such system, it has a continuous interface between physical world and the virtual world for greater convenience and flexibility in user interactions and communications.

2.2 Embedded Robot: Embot

Embot is characterized by ambient intelligence for calm sensing, information processing and communicating in order to integrate assorted sensor information and understand the environmental situation. Its main role is to authenticate various types of robots and robot users. Embot also possesses data-mining ability to collect information about human behavior, status, relationships and environmental conditions such as weather, temperature, etc. Since Embot is implanted in the environment or in Mobot, it allows undemanding communication between robots and humans to improve security, private and public resource management extensively in homes and offices, respectively.

2.3 Mobile Robot: Mobot

Mobot offers a broad range of physical services for general users and specific functions within a specific U-space. Motion and manipulation are key properties of Mobot, which can be implemented in various platforms such as wheel-type or biped to provide a broad range of services such as personal and public services, in-house services and field services based on behavioral intelligence. Mobot communicates continuously with Sobot in order to provide practical services based on commands given by Sobot using Embots' information. Alternately, it serves Embot as a platform for data gathering.

2.4 Middleware

Middleware allows communication within and among Ubibots, which are developed on a variety of platforms using a variety of network interfaces and protocols. Middleware offers common interfaces to Ubibots, thereby making it convenient to componentize the devices and to manage them. It enables the system to make an offer of service irrespective of the operating system, geometric location and type of interface.

3 Multi-Layer Architecture of Ubiquitous Robot

Ubibots in U-space should be interconnected. The properties of Ubibots can be different in many ways because they are developed based on various architectures and platforms with various communication protocols. Thus, software agents should contain all hardware information of the other Ubibots in advance. However, this is practically impossible and rarely possible to expect both modularity and scalability.

As middlewares for distributed systems generally enable accessing heterogeneous devices in a common way, they are good for an application in a static environment with fixed devices [20]. However, Sobots should be able to move into the other environments through ubiquitous networks, where the environments may consist of totally different types of devices with different geometric configurations. Since these information are hardly obtainable for Sobots as mentioned above, we have added an additional abstraction layer, 'Context Provider Layer'. Sobots can query contextual information of environment such as the position of chair or user through the context provider layer without accessing the devices directly.

Overall structure of Ubibot system using the proposed multi-layer architecture is shown in Fig. 1, which is composed of five layers: *Software Agent Layer*, *Sobot Management Layer*, *Context Provider Layer/Task Scheduler Layer*, *Device Management Layer* and *Physical Layer*. Embots and Mobots in 'Physical Layer' physically operate within the ubiquitous environment, where Sobots in 'Software Agent Layer' transfer into various robot platforms and execute a requested service. Proposed middleware consists of 'Sobot

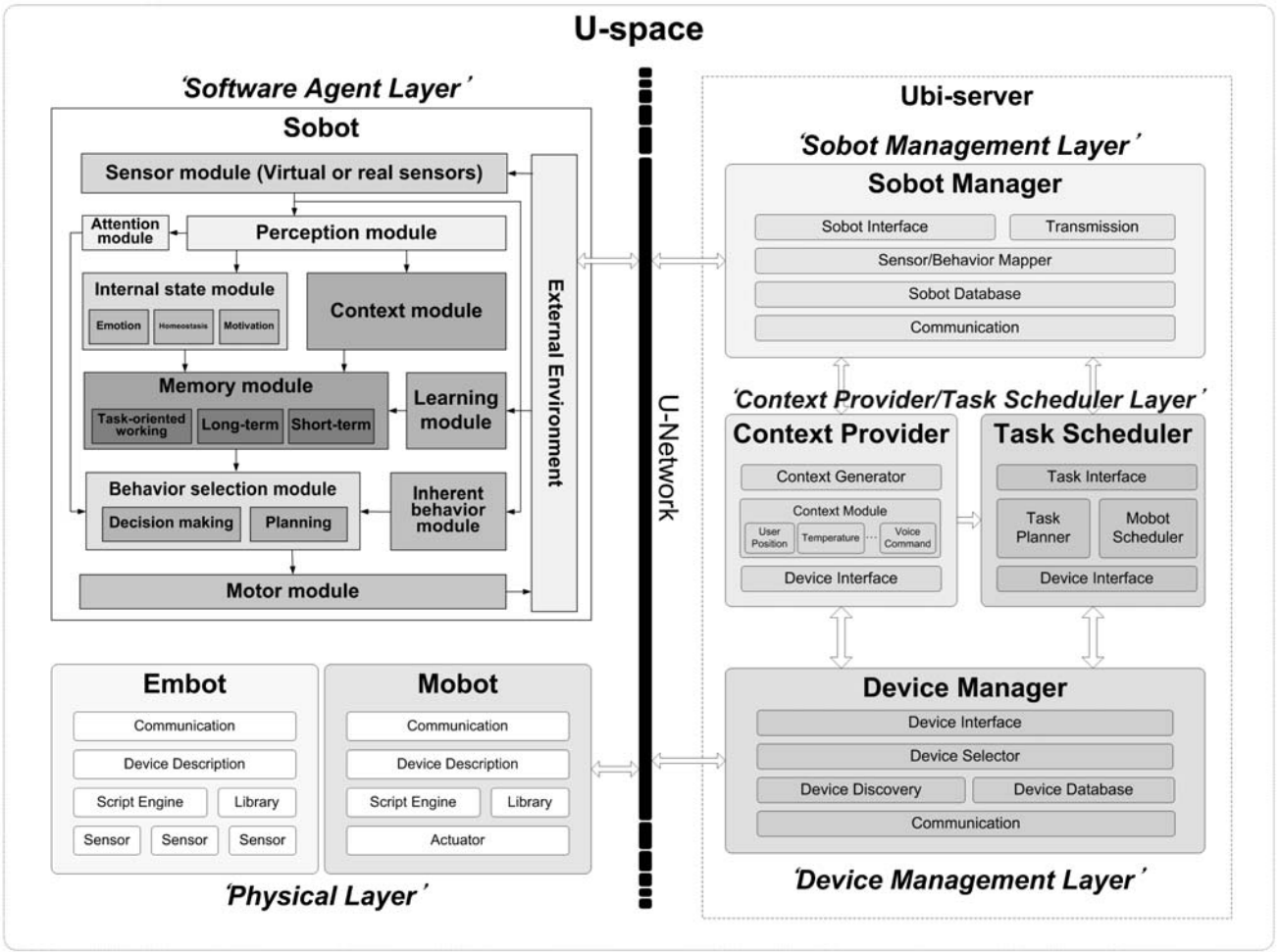


Fig. 1 Overall structure of ubiquitous robot system

Management Layer', 'Context Provider Layer/Task Scheduler Layer' and 'Device Management Layer', which is designed to operate in the U-space. Ubi-server manages all components and devices in U-space. Note that every component interacts with others through the Ubi-server.

If the robot is developed as a stand-alone system in which all the functionalities are installed, the architecture may not be needed to execute a requested service. However, there are many cases where one robot may not effectively perform the required service. Also, in cases where several robots in various platforms are formed to hold various functions, the communication protocol between each robot is different. In order for a Sobot to deal with such issues, it should be aware of all information related to Mobot's protocols and executable functionalities, but this is difficult to achieve in reality. Proposed architecture provides a common interface and possesses an ability to select a most appropriate Mobot for executing the requested service.

3.1 Physical Layer

'Physical Layer' includes both Embots and Mobots. As mentioned in Sect. 2.2, Embot processes raw sensor data to generate meaningful information, which is used to decide current context. Embots and Mobots send their device descriptions to 'Device Manager' to register themselves. In order to cope with platform independency and reprogrammability, Embots and Mobots embed a script engine and libraries to execute any given procedure.

3.2 Device Management Layer

In 'Device Management Layer', the 'Device Manager' manages devices based on the device descriptions of Embots and Mobots, which are stored in 'Device Database' and composed of the hardware properties including the type, number and parameters, localizations, communication protocols, etc. Since 'Device Manager' provides common interfaces to access Embots and Mobots, it is possible to access them

without knowing their hardware specific information including communication protocols. 'Device Discovery' module monitors ubiquitous network and requests Embots and Mobots to register when they are connected to the network. If the registration request is received from the ubiquitous server (Ubi-server), corresponding Embots and Mobots send their device description and then the registration process becomes completed.

3.3 Context Provider/Task Scheduler Layer

'Context Provider Layer/Task Scheduler Layer' deduces the current situation by integrating all types of sensor information. If the situation cannot be deduced due to the lack of sensor information, it transmits a locomotion command to Mobot so that Mobot can collect sensor information actively. Once Sobot decides its behavior, it plans detailed operations and procedures to be exhibited through Mobot. In addition, the main role of 'Context Provider Layer/Task Scheduler Layer' is to rank the priority of tasks when one Mobot is in a position to perform many tasks at once.

3.4 Sobot Management Layer

To incorporate Sobot, Embot and Mobot consistently as Ubibot, 'Sobot Management Layer' is needed to arbitrate different protocols among them. Since Sobot has virtual sensors and carries out its behaviors in a virtual environment, it has physical limitations to serve human beings in real situations, to interact physically and naturally with them, or to make their own physical behaviors. In order to overcome these physical limitations, Sobot must utilize Embot and Mobot that have sensors, physical body and actuators to execute behaviors through a real motor system [21].

When the context information is given from 'Context Generator', 'Sobot Manager' sends this information to Sobot. It also manages the transmission of Sobots when they are requested to move into other Mobots or virtual environment. Since Sobot is a virtual agent, the 'Sobot Manager' should provide a mapping relationship between virtual sensors/behaviors to physical sensors/behaviors of Mobot, which are converted in 'Sensor/Behavior Mapper'.

3.5 Software Agent Layer

As mentioned in Sect. 2.1, Sobot in 'Software Agent Layer' analyzes the given context information received from 'Sobot Manager' and decides which services should be provided to users. In order to provide proper services even if the environment changes, Sobot can move into any hardware device in the particular environment through network. Note that there can be many Sobots depending on the purposes and functions, where each of them basically makes a decision based on context information. For this purpose, Sobot has an internal architecture, which consists of ten modules, as shown in Fig. 1. It has three internal states: 'Emotion', 'Homeostatis', and 'Motivation'. Sobot's memory is composed with three types of memory: 'Task-oriented working memory', 'Long-term memory', and 'Short-term memory' [18].

3.6 Overall sequence

Figure 2 shows the sequence of Ubibot registration and context generation in the multi-layer architecture. The sequence is explained in the following:

(i) Embot/Mobot/Sobot Registration: Embots and Mobots register themselves to device manager and Sobots to Sobot manager (Register). Device manager requests device descriptions to Embots and Mobots (GetDeviceDescriptions()). Embots and Mobots send device descriptions to the device manager (DeviceDescriptions).

(ii) Context Information Request: When Sobots need context information to recognize current situation or user's request, sobots request context information to Sobot manager (GetUserLocation()). Then the Sobot manager requests context information to the context provider (GetUserLocation()).

(iii) Context Information Generation: The context provider reauests and obtains a list of available devices from the device manager (QueryDevice() and SetOfAvailableDevices). The context provider sends a procedure to Embots in the list (UserLocalizationProcedure()) through the device manager and the Embots provide sensor information by executing the given procedure (UserLocalizationProcedureResult).

(iv) Sensor Locomotion Command and Context Information: When all sensor information are gathered, the context provider generates context information and sends it to the Sobot manager (UserLocationContext). If more sensor information are needed, sensor locomotion command is sent to the task scheduler (SensorLocomotionCommand) to drive the Mobot to get the requested sensor information by using its Embot. The generated context information is sent to Sobots which requested the context information (UserLocationContext). Task scheduler decides which Mobot should move and sends locomotion commands to the selected Mobot (SensorLocomotionCommand).

4 Simulations and Experiments

Proposed multi-layer architecture addresses problems of interoperability between different hardware and software platforms in an Ubibot environment in order to offer services

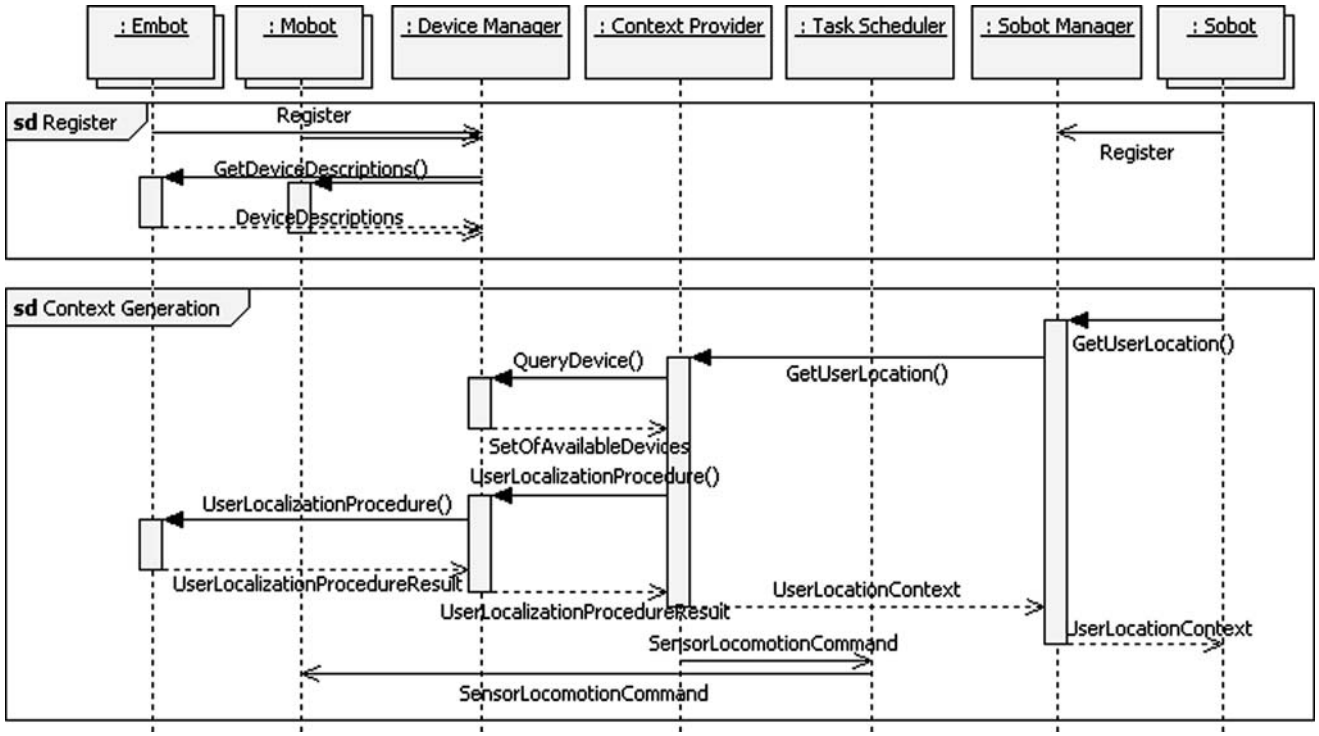


Fig. 2 Sequence diagram of device registration and context generation

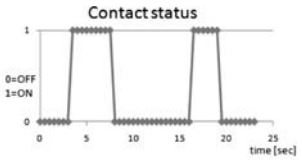
to the user regardless of space or time constraints. In order to verify the feasibility of the proposed architecture, a virtual U-space was simulated with five types of Embots and one Mobot by integrating sensory information to generate context information using two types of Sobots. Also, actual U-space was implemented by synthesizing embedded, software and hardware robots. Note that this paper shows the general feasibility of the proposed concepts, instead of providing a specific practical service.

4.1 Simulation results

To show the feasibility of multi-layer architecture, Ubibot simulator was developed to demonstrate that context information of Embot and Mobot, which were registered within Ubi-server, can be managed and necessary sensor information of any devices can be obtained without knowing prior knowledge of Embot and Mobot. The simulator was made up of five Embots, two Sobots and one Mobot.

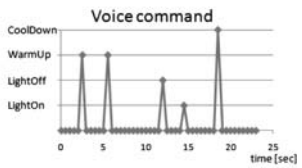
Five simulated Embots—vision Embot, contact-sensing Embot, position Embot, sound Embot and temperature-measuring Embot were used, where sensory information of five Embots are shown in Fig. 3. Vision Embot acted as a camera with an image virtually created from the virtual U-space. In Fig. 3(a), the left one shows global view whereas the right one shows the view from vision Embot. Square box in the left one represents vision Embot and two circles are objects in the virtual U-space. Contact-sensing Embot gave

(a) Vision Embot (square box) and virtual image of objects (circles)

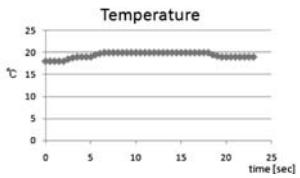


(b) Contact-sensing Embot

(c) Position Embot (grid), object (circle) and detected position (dot)



(d) Sound Embot



(e) Temperature-measuring Embot

Fig. 3 Sensor information of five Embots in simulator

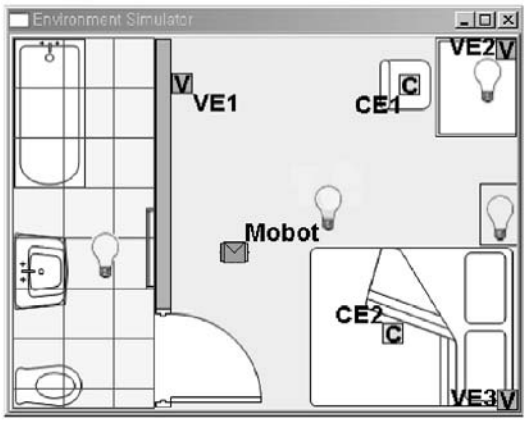


Fig. 4 Virtual environment (VE: Vision Embot, CE: Contact-sensing Embot, grid: Position Embot)

an output 1 if any object was in contact unless 0 as shown in Fig. 3(b). In Fig. 3(c), the left one with grid region shows the simulation environment covered with position Embot whereas the right one shows the detected position of an object within this region. Sound Embot was able to detect four simple commands: 'LightOn', 'LightOff', 'WarmUp' and 'CoolDown' (Fig. 3(d)). Temperature-measuring Embot sensed the temperature of virtual U-space (Fig. 3(e)).

Two Sobots used context information to make decisions on behavior selection. One was light-control Sobot and the other one was temperature-control Sobot. Light-control Sobot decided which light should be turned on or off using both user position and voice command. If user went to bed and did not move for awhile, light-control Sobot turned all lights off. Temperature-control Sobot managed the temperature, where it either increased or decreased the temperature of U-space by one degree when the user requested 'WarmUp' or 'CoolDown' command, respectively.

The size of virtual U-space was 8 m × 6 m, which was divided into a bathroom and a living room as shown in Fig. 4. The virtual U-space was made up of one Mobot, three vision Embots (VE), two contact-sensing Embot (CE), one temperature-sensing Embot, one sound Embot and one position Embot which covers the whole area of bathroom. There were four light stands and a temperature controller. Two Sobots, light-control Sobot and temperature-control Sobot, continuously interacted with Ubi-servers, where Embots and Mobot provided a proper service when user requests a service. Ubibot system recognized the user location by using five types of Embots in order to turn on or off a light located closest to the user and to control room temperature.

Three context information were obtained from the sensor information of Embots. Context of user position was provided by calculating the exact position of users or estimating the probability occupancy map [22]. If the sensory information was not enough to calculate the exact location, a locomotion command was sent to Mobot scheduler

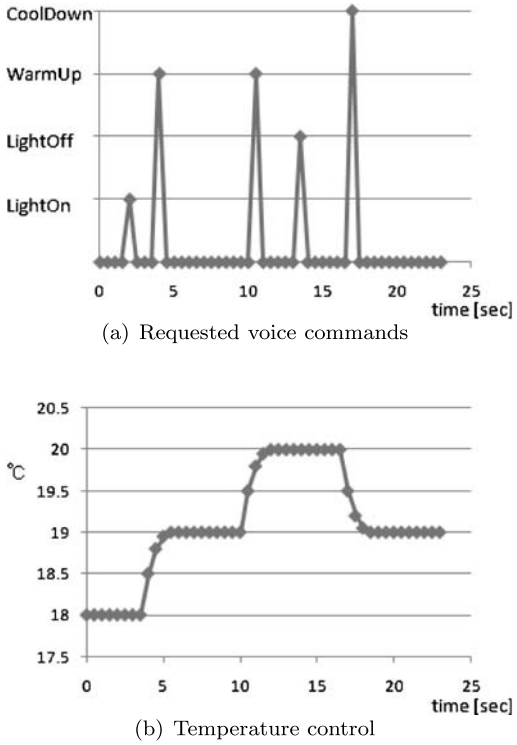


Fig. 5 Simulation results of multi-layer architecture

to move Mobot to obtain more sensory information. Temperature context and voice context were directly generated from the sensory information.

The simulation results showed that the context information was well created from Embots' sensory information as shown in Fig. 5. When the user requested 'WarmUp' or 'CoolDown', sound Embot recognized it and temperature-control Sobot increased or decreased the room temperature by one degree as shown in Fig. 5(b). When 'LightOff' was requested, vision Embots calculated a user position and then light-control Sobot turned off the light, which was located closest to the user position. Since Embots, Sobots and Mobot continuously interacted with Ubi-server, the requested services were provided properly.

4.2 Experiment Results

The implementation of ubiquitous robot system includes a continuous connection via middleware within a network as well as Sobot, Embots and Mobot. Since this paper looks from the central aspect of overall architecture, implementation of basic properties was provided to show its feasibility. Ubibot was developed to demonstrate its paradigm.

The experiment environment was made up of one Sobot, three vision Embots, one position Embot and one Mobot. Sobot called Rity, a dog-shaped 3D virtual creature with 12 degrees of freedom as shown in Fig. 6, was developed to fulfill the requirements for an artificial creature. The artificial

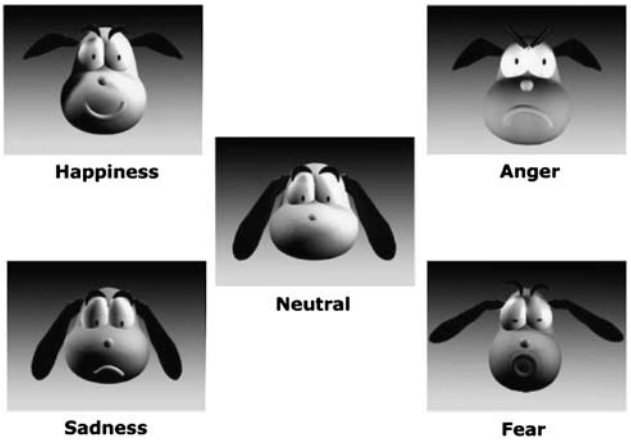


Fig. 6 Pet-type Sobot, Rity

dog, Rity, was made up of a set of computerized DNA codes, which has its own personality and ability to reproduce and evolve as a distinct species [18, 23, 24]. Rity had various virtual sensors such as a light sensor, a sound sensor, a thermal sensor, a vision sensor, a gyro sensor and an equipped timer.

Rity has a complex internal architecture with 14 internal states. It has 47 perceptions, exhibits 5 facial expressions and is able to conduct 77 behaviors. Rity holds several essential internal state components like humans such as motivation, homeostasis and emotion. It is an intelligent software robot that lives inside the virtual environment, but interfaces with the physical environment through the Embots in the network. The virtual environment was matched up to the physical environment by using RFID tag floor covering with position Embot, which will be introduced in the next paragraph.

There were various Embots to process sensor information. We considered five types of Embots in this paper: vision Embot, contact-sensing Embot, position Embot, temperature-measuring Embot and sound Embot. Among five Embots, vision Embot and position Embot were physically implemented for experiments. Vision Embot was able to detect objects according to their colors and recognize the faces of people using USB camera. It was developed for Sobot using a bottom-up feature-based detector and Principal Component Analysis (PCA)-based face recognizer [25].

Position Embot recognized the positions of objects using a 13.56 MHz reader module with a PCB type antenna and a RFID tag floor covering a sectioned arrangement of square tags. 256 RFID tags covered the floor, which are placed with the sectioned arrangement spaced by 10 cm gaps in the total area of 1.5 m × 1.5 m covering a sectioned arrangement of square tags [26]. Position Embot transmitted the detected positions of the robot in Cartesian coordinates to middleware once every tenth of a second.

A mobile robot type Mobot, Mybot, used a differential drive platform powered by DC motors. It is 31 cm × 21 cm × 42 cm and weighs 12 Kg. It has an onboard Pentium III 850 MHz computer, six Polaroid 6500 ultrasonic sensors, and Tablet PC to control and operate. A USB camera was provided to enable the operation of the vision Embot. The platform could move with a top speed of 70 cm/s and a peak acceleration of 300 cm/s^2. In addition, it had two arms with grippers at the end to express its emotion from the five behavior modules in Mybot [27].

Once Embots, Mobot and Sobot appeared on the ubiquitous network, they registered themselves to Ubi-server sending their device descriptions. Sobot tried to find user's position to interact with the user and requested his position to Ubi-server. 'User Position Context' module was invoked and the module requested a list of available vision Embots and position Embots. Context of user position was calculated or estimated with the sensor informations which were obtained from the vision Embots and the position Embots.

When the user was in his office, Sobot appeared on the screen of the user's personal computer. Once the user vacated the office and moved into the hall, Sobot was transmitted into Mobot which was the nearest device from the user. Having tracked and followed its user, the Sobot then remained on the screen to allow him to have further interactions. Figure 7 shows the sequence of the experiment following the overall sequence described in Sect. 3.6. When context information of user's position was provided to Sobot, Sobot was transmitted into a proper device to provide seamless interaction with the user.

Context information were generated continuously utilizing Embot-attached Mobot with the proposed middleware without directly consulting the user or Sobot. Using the generated context information, Sobot moved into the nearest device from the user without prior information of the devices in U-space.

In the implementation, Sobot, Embots and Mobot were connected in network. They will co-exist with humans to provide seamless, calm and context-aware services at any place and anytime through the Ubi-server. Distributed Embots ensure Sobot to possess context-aware capabilities and to overcome spatial limitations. Embots authenticate and integrate many sensors to comprehend the current environmental situation. Mobots are capable of providing service requests in the physical domain. Lastly, middleware is an important component to communicate within and among Ubibots using a variety of network interfaces and protocols.

5 Conclusion

This paper proposed a multi-layer architecture consisting of five layers, to solve the problem of interoperability between

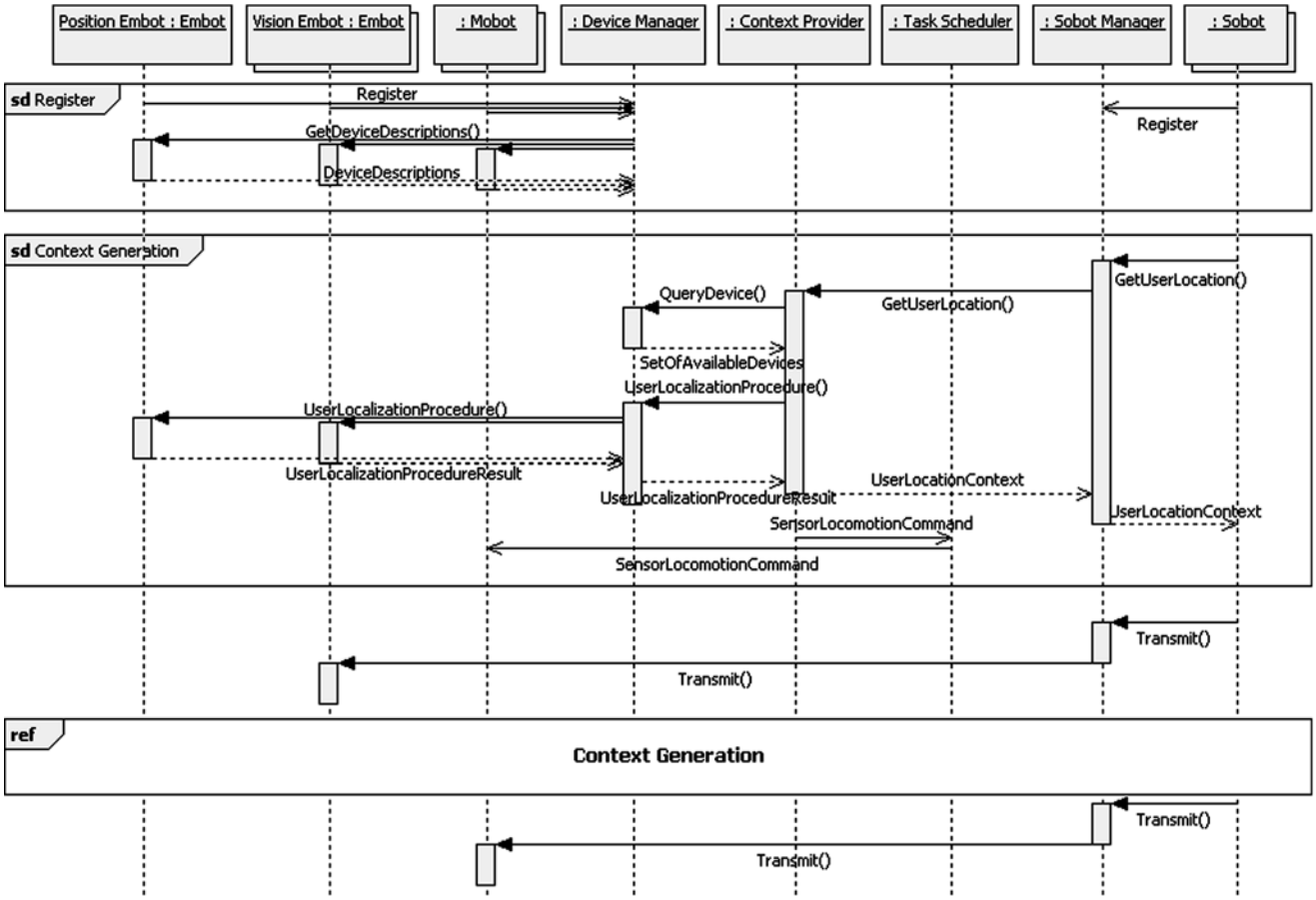


Fig. 7 Sequence diagram of the experiment

different hardware and software platforms in a Ubibot environment. Separating context generation and software agents, Sobots were able to provide proper services to users when they moved to other environments without knowing which sensors were placed in the environment. In order to show the feasibility of the proposed architecture, a U-space in a virtual environment was simulated.

The implementation of the virtual U-space allowed Sobot to acquire context information from Embots, which were made up of virtual sensors such as cameras, RFID tags and contact sensors. The feasibility of Ubibot system were shown by implementing a Sobot, which was a pet-type 3D artificial creature Rity, vision Embot, position Embot and Mybot. The integration of Sobot, Embot and Mobot fully understood the user's movement within a networked environment.

As a further work, more experiments in a U-space in which many more Embots and Mobots are interconnected through ubiquitous network are necessary to realize all the properties of Ubibot. Problems of ensuring privacy and protecting property in Ubibot should be solved by using solutions from the computer technology.

References

1. Weiser M (1993) Some computer science problems in ubiquitous computing. Commun ACM. doi:10.1145/159544.159617
2. Weiser M (1999) Some computer science issues in ubiquitous computing. ACM SIGMOBILE Mobile Comput Commun Rev. doi:10.1145/329124.329127
3. Murphy RR (2000) Introduction to AI robotics. MIT Press, Cambridge
4. Breazeal C (2004) Social interactions in HRI: The robot view. IEEE Trans Syst Man Cybern Part C: Appl Rev. doi:10.1109/TSMCC.2004.826268
5. Luo RC, Su KL, Shen SH, Tasi KH (2003) Networked intelligent robots through the Internet: Issues and opportunities. Proc IEEE. doi:10.1109/JPROC.2003.809198
6. Ha YG, Sohn JC, Cho YJ (2005) Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic Web services technology. IEEE/RSJ Int Conf Intell Robots Syst. doi:10.1109/IROS.2005.1545016
7. Kim JH (2004) Ubiquitous robot. In: Reusch B (ed) Computational intelligence, theory and applications. Springer, Berlin, pp 451–459
8. Kim JH, Kim YD, Lee KH (2004) The third generation of robotics: Ubiquitous robot. In: Proceedings of international conference on autonomous robots and agents, pp 1–7
9. Levis P, Culler D (2002) Mate: A tiny virtual machine for sensor networks. ACM SIGOPS Oper Syst Rev. doi:10.1145/635508.605407

10. Hadim S, Mohamed N (2006) Middleware: Middleware challenges and approaches for wireless sensor networks. Distrib Syst Online, IEEE. doi:10.1109/MDSO.2006.19
11. Yao Y, Gehrke J (2002) The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Rec. doi:10.1145/601858.601861
12. The Object Management Group (1999) The common object request broker: Architecture and specification, Revision 2.3
13. The Object Management Group (2004) Platform independent model and platform specific model for super distributed objects specification, Version 1.0
14. Ranganathan A, Campbell RH (2003) A middleware for context-aware agents in ubiquitous computing environments. Middleware 2003: ACM/IFIP/USENIX international middleware conference. doi:10.1007/3-540-44892-6_8
15. Kim JH, Park IW, Jeong IB, Lee KH (2008) Ubiquitous robot for integrated network services. J Harbin Inst Technol (New Ser) 15(Sup 2):2–5
16. Kim YH, Cho SH, Choi SH, Kim JH (2007) Software robot in a PDA for human interaction and seamless service. In: The 16th IEEE international symposium on robot and human interactive communication. doi:10.1109/ROMAN.2007.4415223
17. Kim JH, Lee KH, Kim YD, Kuppuswamy NS, Jo J (2007) Ubiquitous robot: A new paradigm for integrated services. In: 2007 IEEE international conference on robotics and automation. doi:10.1109/ROBOT.2007.363904
18. Kim JH, Lee CH (2008) Multi-objective evolutionary generation process for specific personalities of artificial creature. Comput Intell Mag IEEE. doi:10.1109/MCI.2008.913368
19. Kim JH, Cho SH, Kim YH, Park IW (2007) Two-layered confabulation architecture for artificial creature's behavior selection. IEEE Trans Syst Man Cybern Part C: Appl Rev. doi:10.1109/TSMCC.2008.2001576
20. Bernstein PA (1996) Middleware: A model for distributed system services. Commun ACM. doi:10.1145/230798.230809
21. Kim TH, Choi SH, Kim JH (2007) Incorporation of a software robot and a mobile robot using a middle layer. IEEE Trans Syst Man Cybern Part C: Appl Rev. doi:10.1109/TSMCC.2007.905850
22. Isla DA, Blumberg BM (2002) Object persistence for synthetic creatures. In: AAMAS '02: Proceedings of the first international joint conference on autonomous agents and multiagent systems. doi:10.1145/545056.545132
23. Kim JH, Lee KH, Kim YD (2005) The origin of artificial species: Genetic robot. Int J Control Autom Syst 3(4):564–570
24. Kim JH, Lee KH, Kim YD, Park IW (2006) Genetic representation for evolvable artificial creature. In: IEEE congress on evolutionary computation. doi:10.1109/CEC.2006.1688545
25. Jang JS, Kim JH (2005) Two-layered face detection system using evolutionary algorithm. In: The 2005 IEEE congress on evolutionary computation. doi:10.1109/CEC.2005.1554868
26. Hahnel D, Burgard W, Fox D, Fishkin K, Philipose M (2004) Mapping and localization with RFID technology. In: The 2004 IEEE international conference on robotics and automation. doi:10.1109/ROBOT.2004.1307283
27. Han KH, Kim S, Kim YJ, Kim JH (2001) Internet control architecture for Internet-based personal robot. Auton Robots. doi:10.1023/A:1008941101455