Network-based Robots and User Interaction Framework

Dong Nguyen To, Do-Ik Kim, Bum-Jae You [a] and Sang-Rok Oh

***Abstract*— In this paper, a network-based service architecture (NBSA) for Ubiquitous Robotic Companion (URC) has been proposed for controlling network-based robots in ubiquitous network environment. By allocating several service servers and sensors in the environment and connecting them with the robots through wireless connection, a network-based robot system can be created. NBSA provides a core program for developing this network-based robot system. Using a multi-agent framework with some service organization extension and coordination agents, behaviors of the robot can be created, added and removed dynamically by users via a simple scripting method. Experimental results show that NBSA provides an extendable behavior control system in the network environment and it decreases the development time for adding new services. Using this architecture, role of a network-based robot, method for coordination and working scenario can be changed easily by users without retouching the core program.**

I. Introduction

Network-based robots are made to serve and assist humans by imitating and providing human-like actions and behaviors based on service contents and intelligence software in external servers outside of the robots. So, they have a large set of basic behaviors and can perform many different tasks by using arms, hands, and mobility. Controlling a system with many degree-of-freedoms and sensors, many real-time sensor data streams consumes a lot of CPU power. Due to the reason, not much space is available for complicated high-level services such as face recognition, gesture recognition, object recognition, reasoning and so on. Therefore, using services on the external servers and connecting them with the robots in an extendable network-based architecture is a good solution. Such a network-based architecture is essential in the sense that a robot becomes a partner and an assistant of human beings in ubiquitous environment, where the external sensors or even household network enabled devices can be connected together to create an intelligent environment.

The architecture is requested to provide functions that a robot can explore external services for its goal and users can explore the robot with the added ability provided by the surrounding servers. The services can be added, removed or changed on the surrounding nodes frequently while embedded control programs of the robot can not be changed frequently. So, the architecture is expected that the robot is equipped with a relatively stable embodied program. Whenever a new service is added, the system should be able to detect this new service and coordinate this new service with existed system. The human-robot interface system should be able to provide all available services for accessing by other partners or users.

The coordination is done using pre-defined rules. In some cases, knowledge discovery and exploration among networked nodes need to be considered. The architecture therefore needs to support a knowledge description language and a reasoning system. Moreover, one network-based robot sometimes is expected to play a different role such as role of a man, a woman, a kid, a butler, a house maid, etc. With each role, the behaviors and/or voices should be changed accordingly. The architecture also needs to provide a solution to do it simply without reprogramming the coordination system in other servers.

Users explore the system using their network access equipment such as computer, notebook and PDA. The robot system sometimes may be offline for battery charging or due to network problems. Users are, however, busy and hope to access the system as fast as possible. The architecture should provide a method to deal with this offline situation. Moreover, users normally are not robotic experts so they do not know exactly what to do. So, a graphical user interface with detail explanation and/or a simple scripting language for scenario, role and coordination describing are very useful.

A client server model, where each robot requests the services to external servers, can be a solution but it is not scalable. Whenever a new service is added, the client code on each network-based robot should be modified to access and to explore the service. Moreover, offline problem is difficult to solve in client-server model. Considering the above requirements, multi-agent technology is a good choice to design the system.

In this paper, there is proposed a Network-Based Service Architecture (NBSA) for robots to deal with problems in network environment. Also, a coordination method is proposed to explore coordination scripts, service agent and multi-agent management system. Using the coordination method, services can be added from service servers without any changing in embedded software of the robots. Behaviors, which are created by the coordination, can be detected, explored and customized by users and the robots using the proposed NBSA.

Next, a role-based method, which helps users to define the role of a robot, is proposed. Using this method, a robot may change all its real implementations of behaviors without affecting the coordination script. The same network-based robot can play a role of a man, a woman, or a kid depending on the profile that can be selected by users at the scene. A service access method is proposed by using

Manuscript received January 31, 2008.

D. N. To, D. -I. Kim, B. –J. You, and S. –R. Oh are with Center for Cognitive Robotics Research in Korea Institute of Science and Technology (KIST), Haweolkok 39-1, Seongbuk, SEOUL 136-791, Korea (corresponding author to provide phone: +82-2-958-5760; fax: +82-2-958-5749; e-mail: ybj@kist.re.kr).

slave agents, agent migration and multi-agent system. Using this method, users can access the system, receive service interfaces and explore it at a highly available level.

In order to evaluate the proposed architecture for network-based robots, a number of experiments on a humanoid robot, *MAHRU*, have been conducted successfully by adopting JadeLeap middleware (Bellifemine, 2003) and FIPA standard (Foundation for Intelligent Physical Agents http://www.fipa.org.) on a system, which consists of a PDA, a network-based humanoid named MAHRU developed at KIST, external service servers, and other mobile robots. An extension of Jess rule engine (the Rule Engine for the Java Platform http://herzberg.ca.sandia.gov/jess/) and corresponding implementation classes are developed for describing and implementing coordination. Experimental results show that the proposed framework decreases the development time, decreases the time for network connection of robots and user devices. Most importantly, it provides the extendable ability for our humanoid system and gives users the way to easily modify behaviors of the humanoid that can explore the service servers located in the network.

II. Related Works

Recently, there are some researches on the distributed architecture for network robotics. Lee, J. et al (Lee, 2004) proposed a Robot Software Communication Architecture (RSCA) that explores real-time CORBA middleware to provide a development framework for a distributed control system. RSCA provides a real-time, object-oriented based method for developing the service in a distributed environment. The method is good for the cross-platform and cross-language development process. It comes together with several standard operations for deploying distributed component-based robot application such as installation, start and stop options. However, software control system should be modified when a new service is installed for the first time. Moreover, all main development tasks are done on the service server side. Hans, Utz. et al (Hans, 2002) introduced another distributed architecture named Middleware for Mobile Robot (MIRO). MIRO also uses CORBA middleware for distributed cross-platform development. MIRO defines the layer architecture including device layer, service layer, and class framework. These layers provide the standard services for robots. MIRO could be extended for supporting external services. However, MIRO could not address the service discovery and coordination problem in offline situation.

In order to coordinate network devices and robots, Baker, D. I. Et al (Barker, 2004) proposed a framework for dynamic distributed architectures. This framework uses task directed method to coordinate distributed systems. It explores a module pool and resource server to create the task controller outside of the standard module. Service discovery and coordination are handled using dynamic loadable share object library. It solves problems of reconfiguration and multi-robot coordination. However, it did not provide an abstract layer for communication and human robot interaction.

Wu, Chao-Lin. et al (Wu, 2004) employed the mobile agent to provide the distributed control architecture for home automation system. This architecture is based on mobile agent to transfer the message agent around the system and coordinate with the functional agent in each host. It addresses the human system interface and task coordination problems. However, transferring message agent for collecting data during the coordinating process causes latency and reduces performance. In our method, only the interface is migrated through network. The coordination is done by a coordination agent, which implements a task description script.

Ha, Young-Guk. et al (Ha, 2005) use semantic web technology for service-oriented integration of networked robots with ubiquitous sensors and devices. This method explores the web ontology language and hierarchical task network (HTN) method in reference (Nau, 2003) for defining the coordination of each node. This approach is good in coordinating robot and environment without any pre-defined information. However, the service management and exploration in offline situation is not solved. Instead of using HTN we directly make scripting of primitive tasks by extending the Jess language. The corresponding implementation library of all standard humanoid behaviors, which can be invoked from Jess script, is also supplied in our architecture.

Dong To Nguyen et al (2005), also did develop a framework for human-robot interaction in network environment. The Virtual Directory Facilitator (VDF) is used in this framework as a method for coordinating mobile robots, distributed sensors, users in different situation. However, a mobile robot can perform only limited tasks so the coordination is mainly for gathering the information from environment for path planning or localization. More complicated robots such as humanoid robots or torso robots with wheel-based mobility can perform many types of tasks and requires coordination with many different types of servers in different situation. Therefore, an easy-to-use scripting method is required for describing the scenario, coordination flow, and role of each robot. The proposed NBSA includes an improved VDF architecture while scriptable role-based systems and coordination methods provide more flexibility for humanoid controlling system development.

III. Proposed Architecture

A. System overview

Fig. 1 shows a physical view of our system. A network-based robot is surrounded by several service servers, which provide external processing services such as face recognition service or gesture recognition service. Normally, these service servers receive requests and data from the robot and send back the analyzing results when it is necessary. The number of service servers is changeable. Robots, users and developers can detect the new service servers and explore them for their purposes using our network-based architecture.

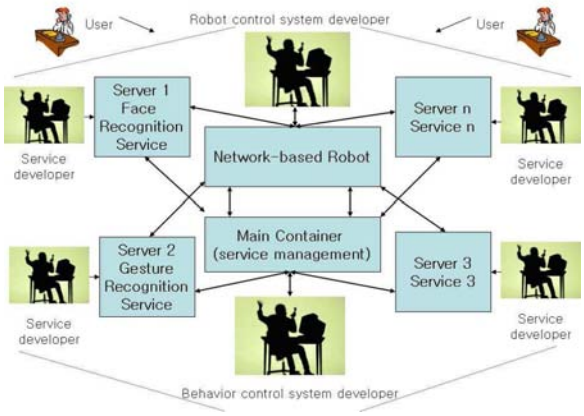


Fig 1. Overall system

For developing of this system, there are three groups of developers, who can explore the proposed architecture in different manner to ease their process of creating, exploring and monitoring a service.

Robot program developers are those who develop the controlling system, primitive behaviors, user interface and exporting information such as sensor data stream from hardware layer. These developers need not to care how many external servers exist and how to coordinate them. They just simply develop and provide the Application Program Interface (API) that can be called by some instructions from behavior level.

Behavior control developers are those who check primitive behaviors, external services and determine which services can be created using these primitive behaviors and current existing services.

Service developers are those who develop external services that can use the exporting data from the robot. In order to use information from the robot, these developers should also develop the coordination agent to register the exporting data and communicate with the service. If the service provides the interface for users, these developers should also develop this interface.

NBSA provides tools and core programming codes for each group of developers to do their tasks. NBSA is designed for two groups of users. Normal users should be able to access the system using its user agents. Normal users select the behaviors in the list of behaviors. Then they gain access to the interface of this behavior and interact with this interface if it is necessary. Expert users even could define some rules that change the functions of the whole system. Using the role function and coordination script, which will be described in the next section, expert users can change the role of a robot and all of its behaviors in each situation. Expert users can also be able to change the way of exploring the service servers in the network environment.

B. Overall architecture

Fig. 2 shows the proposed network-based service architecture with 4 blocks: Robot Block, Main Container Block, Service Server Block and User Container Block.

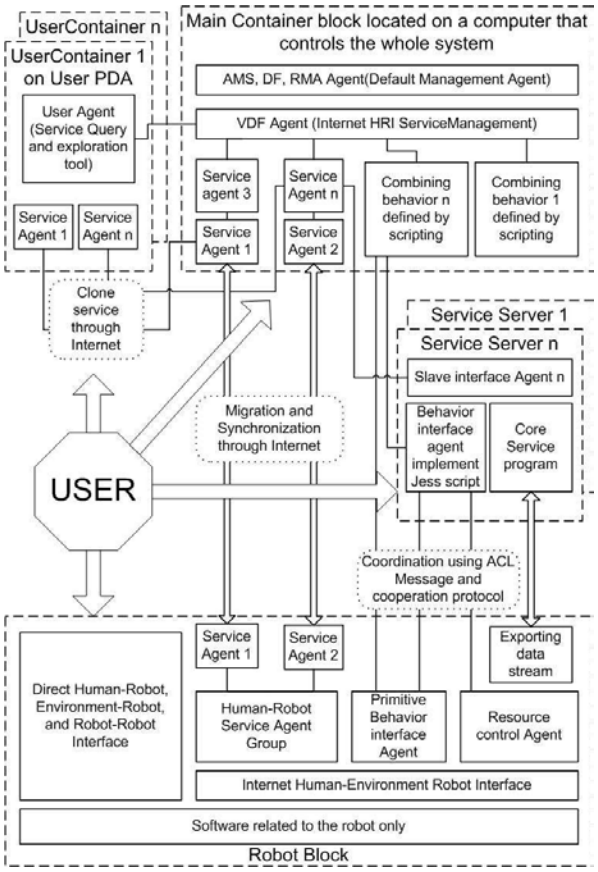


Fig 2. Overall structure of NBSA

Robot block

To be a partner of a multi-agent system, the control software located on a robot is redesigned into two parts: the control part and the interface part. All behaviors and services in the interface part are arranged as agents by considering only the network-based services. Human-robot interface services require human interaction. With these services, robots create slave agents and migrate them to the main container. These services require no pre-defined algorithm and configuration. Users can send command and work with the services by using their graphic user interface (GUI).

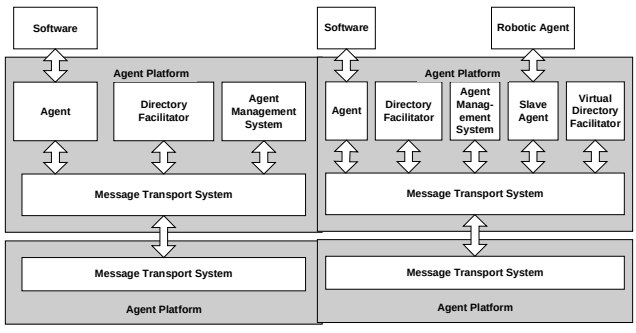
The primitive behavior interface agent is the interface to the behavior that interacts with outside partners. A service or behavior can coordinate with the external service provider and these primitive behaviors by defining a set of rules for processing the coordination information in a task scripting rule base language. Jess rule-base is selected to implement this script because it supports external function invocation and can translate the agent message as facts for its rule based system. By sending messages to the interface agents, partners can invoke a command in a network-based robot system. This command can be called directly through a behavior of the interface agent, or indirectly through some

processing units such as Jess rule-based engine.

Main container block

Main container is a main service management block. FIPA standard is selected to implement the framework. FIPA standard defined a management system using a central control in its main container. In our architecture, the implementation part and the interface part of a service are separated so that the interface part can be moved around network by migrating. Each robot appears in the system as an agent. Each time this agent registers to the agent platform, it creates its slave service agents. Using the mobility feature of the agent platform, slave agents will be migrated to locate on the main container. These agents work as slaves for the robotic agents and service agents. They are created at the time robotic agents and service interface agents appear in the system and remains for a particular period specified by these agents. The slave agents collect information when the robotic agents are offline, get the requirement or interact with users. It can provide complicated graphic interface. Therefore, users with no experience about the robot can interact with it because after being migrated to the user device, it can be used as a normal application program. The slave agents are synchronized with robotic agents when they are online. To control such a slave agent group, a special agent in the agent platform is created. It is called as a Virtual Directory Facilitator (VDF) agent. VDF works similar to Directory Facilitator (DF) in FIPA standard. However, all of these services are provided by slave agents as shown in Fig 3.

VDF is responsible for synchronizing the slave agent and robotic agent, service interface agent. The User Agent (UA) located on the PDA of each user works as a service query-and-display tool. UA can query all the services currently available on a system and then select the service that users want to use. After a service is selected, VDF will clone a copy of the service interface and move it to the container in PDA. After users interact with the interface, all data will be synchronized among interface agent at the service level. VDF will be responsible for finding the time and partners of the synchronization.



a. FIPA Agent Platform b. FIPA with VDF extension

Fig 3. Management architecture of FIPA standard and VDF

This system helps external services and each robot to broadcast its service even after it deregisters from the system temporally. Instead of only giving the name of service, the real interface of the services is given. Users can save the time for interaction with services because they can access the system at any time. Then they can work with the interface in the VDF and explore this service by invoking the interface. The availability of the system increases since the demand of long-term or permanent network connection of the system is reduced and each robot and external services only need to be connected to the system for a short time to migrate its service interface or to synchronize the data.

Besides controlling the interface of services, VDF also control the list of coordination agents that coordinate the humanoid and the service provider. Normally the coordination process is specified by a scripting snippet. When user selects a behavior from VDF, coordination agents will run the corresponding scripting code and the system operates under the coordination rules in the script. VDF should do the synchronization and check each partner in behaviors to assure that all partners are available at the implementation time. If not, it should deactivate the state of a service.

TABLE 1. Role of virtual directory facilitator (VDF)

Order	Role
1	Control the behavior lists including primitive behaviors and new created behaviors
2	Clone, migrate service interface to user container
3	Synchronize data among interface agent
4	Control the data flow in each service

Service server block

Service block is located on the service provider computer. The core service program processes the data received from the robot, analyses and sends back the result to the robot. In order to explore the service, a scripting program is used to coordinate the service and the behaviors of the robot. Users can also directly access the service and customize it using a slave interface agent that can be migrated to the main container and later on to the user container. Table 2 shows the tasks of service server block. The core service interface agent receive request from the robot, pass it to the Jess engine as a fact for ruled-based system. The corresponding action will be carried out according to the rules that are fired in the coordination script.

TABLE 2. Tasks of service server block

Step	Task	Action
1	Register the service	Send register message to virtual directory facilitator.
2	Request input data	Request sensor data as input for the analyzing service
3	Provide service user interface	Create and clone the slave agent for service interface.
4	Add new behaviors	Add new behaviors to the VDF and provide a set of coordination

		rules for each new behaviors

User Container block

Users access the system via an UA located on their PDA. The PDA does not have any information about the robots. UA queries and clones the available services from the main container and users can interact with the services via ACL communication by using the UA.

TABLE 3. Task of user container block

Step	Task	Action
1	Register the user	Send register message to VDF.
2	Request service lists	Send request to the VDF to get list of available service
3	Got service interface	Clone and migrate the selected service to user container.
4	Interact with service	Interact with slave agents and request synchronizing data
5	Request result	View result from previous tasks
6	Create a behavior script	Expert user can create a task description script for action.

C. Coordination with external service servers

Fig. 4 shows a detail of coordination flow of a system that explores a separate coordination script to coordinate a robot and a service. Coordinator agent at the robot and service container should receive the messages from other coordination agents. Then these messages are sent to Jess Engine for processing. Then Jess engine check the rule inside the service or robot behavior script code and send back the corresponding message or invoke the native commands through Java Native Interface if it is necessary.

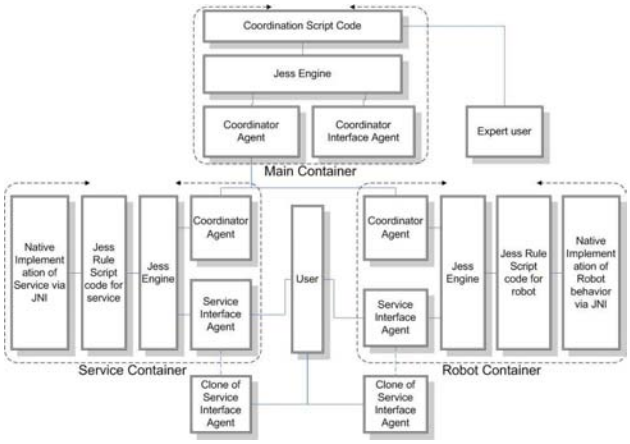


Fig 4. Role of coordination script and service interface agent

With the same coordination script, behaviors of each partner may be changed by accessing the service interface agent that located on the robot container, service container or clone versions of this agent, which have been cloned and migrated to the user container. After users interact with the interface agent, the robot can be changed to play the role of a child, a man or a woman. The same command will be interpret differently depend on the role selected. After changing the role of each partner, the system will behave differently without changing the main coordination script. Coordination script is created by expert users and located in the main container for accessing. In other cases, service servers and the robot can be coordinated directly by using the script located in service servers.

IV. IMPLEMENTATION & EXPERIMENTS

The proposed NBSA is tested with a humanoid, named *MAHRU*, developed by Center for Cognitive Robotics Research in KIST. *MAHRU* is a network-based humanoid developed as one of Ubiquitous Robotic Companion. The appearance is as shown in Fig 5. The height is about 150cm and the weight is about 67Kg. It has 6 DOF for each leg and arm, 1 DOF for waist, 2 DOF for neck, and each hand has 4 DOF. *MAHRU* is equipped with various sensors including a stereo camera system, force/torque sensors, a microphone while it is connected with external service servers in wireless network environment. Now, *MAHRU* can work with external service servers for face recognition, voice recognition, gesture recognition, three-dimensional object recognition, and information service.

Fig 5. Network-based Humanoid “MAHRU”

The experiment is conducted with the User Agent (UA) on a PDA, iPAQ 5450, using JadeLeap. This iPAQ connects to the network via an 11 Mbps wireless connection. The UA is written as an agent in an agent container on Jade Leap platform. In contrast to the conventional robotic control system, this UA does not have any particular information about robots or environments. After registering to the agent platform, the UA can search for the existing VDF, query the VDF on available services. VDF contains the services of robotic agents that are currently online and offline. After

getting the list of services available in the system, UA may send query VDF to request a special service. This service interface is cloned in the main container and migrated to the user container. Users run the interface on the PDA, work with the service interface and leave the system. Later on, the users search the result using this UA.

NBSA shows advantages to the client-server model as describe in Table 4. Offline services can be done manually for each service using client-server model. However, NBSA system helps users to get the GUI of services and work directly with any service in the system without caring how it is synchronized with the real implementation part. With this implementation it also helps to work with the group of robots, or change behaviors of the whole groups by interacting with the interface given by a member of the group.

TABLE 4. Comparison with the client-server control system

Properties	Client-server	Multi-agent with VDF and slave agent
Offline service interface	No	Yes. (Users work with the interface via a clone version of slave agent)
Behaviors of a group can be changed in each environment	No	Yes. Robot interacts with the system to get information and its behaviors can be changed.
Selection of robots	No	Yes (based on service requirements)
Dynamically add, remove services	No	Yes (Using VDF, Coordination agent and Jess script)

V. CONCLUSION

We hereby have proposed a Network-based Service Architecture, NBSA, which is suitable for network-based robot software development of users. VDF and slave agents work together to provide a solution for interacting with services even in offline situation. VDF, the coordinator agents, Jess scripting code and Jess scripting engine work together to coordinate various types of agents - such as slave agents, primitive behavior interface agents and service interface agents - for interaction. Using NBSA, external services can be added and removed without affecting the robot operation and program. A library of standard robot action for a humanoid, which can be called from Jess script, is also provided. Using the core classes and management block of the architecture, developers can concentrate on making the services and their application. NBSA contributes to the network-based robot field by introducing the usage of a multi-agent system with slave agents for services, a VDF agent, external service servers, and coordination agents.

However, NBSA needs to be developed to deal with various problems appear during its operation. Jess script is somewhat difficult for normal users. In the next step, we are developing a simple scripting language for users. For knowledge description, a framework such as Protege (Knowledge description framework, http://protege.stanford.edu/) is being investigated for exploring services and coordinates in ubiquitous environment. NBSA is not yet a complete solution for solving all problems in the network-based robot system but it is a good solution in term of flexibility and extendibility.

REFERENCES

[1] Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G., (2003), JADE - A White Paper, in Journal "EXP - in search of innovation", September, volume 3, pp 6-19.

[2] Baker, D. I., McKee, G. T., Schenker, P. S., (2004), Networked Robotics, a framework for Dynamic Configurable Architectures, in Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan pp. 1768-1773.

[3] Wu Chao-Lin.,Wei-Chen.Wang, nd Li-Chen. Fu, (2004), Mobile Agent Based Integrated Control Architecture for Home Automation System, in Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, pp. 3668-3673.

[4] Ha. Young-Guk, Sohn. Joo-Chan, and Cho. Young-Jo, (2005), Service-Oriented Integration of Networked Robots with Ubiquitous Sensors and Devices Using the Semantic Web Services Technology, in proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems, Alberta, Canada, August, pp. 413-418.

[5] Hans. Utz, Stefan. Sablatnög, Stefan. Enderle, and Gerhard. Kraetzschmar, (2002), Miro - middleware for mobile robot applications, IEEE Transactions on Robotics and Automation, Volume: 18, Issue: 4 , August, pp. 493 – 497.

[6] Lee. J, Park. J.-Y, Han. S. and Hong. S, (2004), RSCA: Middleware Supporting Dynamic Reconfiguration of Embedded Software on the Distributed URC Robot Platform, in Proceeding of the First International Conference on Ubiquitous Robots and Ambient Intelligence (ICURAI), pp. 426-437, Seoul, Korea, December.

[7] Nau. D.S, Au. T. C, et al, (2003), SHOP2: An HTN Planning System, in Journal of Artificial Intelligent Research, Vol. 20, AIAF, pp.379-404.

[8] Dong To Nguyen, Sang-Rok Oh, Bum-Jae You (2005), A Framework for Internet-based Interaction of humans, robots and responsive environments using agent technology, IEEE Transactions of Industrial Electronics, Special Section on Human-Robot Interface, Volume 52, Number 6, December 2005, pp.1521-1529.