Published in ICASE/SICE Joint Conference on Intelligent Control and Systems, , Muju Resort, Korea, Oct. 16, 2002 (hosted by Institute of Control, Automation and System Engineers (ICASE, Korea) and Society of Instrument and Control Engineers (SICE, Japan)

Intelligent Association in Agent-based Ubiquitous Computing Environments

H. Duman, H. Hagras, V. Callaghan, G. Clarke and M. Colley

Department of Computer Science, University of Essex, Wivenhoe Park, CO4 3SQ, Colchester, United Kingdom
(E-mail: hduman@essex.ac.uk)

Abstract: Our living spaces are becoming increasingly populated with infinite numbers of intelligent embedded agents we are interacting with. In ubiquitous computing environments the aim is to support the occupants during their everyday lives and enhance their living conditions. In this paper we introduce such an environment, the iDorm, which has arisen from our work in careAgents, a project supported by the UK-Korean S&T collaboration fund and the EU's Disappearing Computer Initiative eGadgets project. We discuss the research challenges involved, particularly those relating to intelligently associating and configuring large numbers of embedded agents. The paper presents an intelligent association system (IAS) that can overcome problems of existing user-dictated association technologies on multi-embedded agent systems operating in dynamic ubiquitous computing environments. We also introduce the IAS Editor, a tool to create and visualize associations and discuss the high-level multi embedded-agent model, explaining how it facilitates inter-agent communication between heterogeneous sets of agents.

Keywords: Intelligent Associations, Ubiquitous Environments, Multi-agent Systems, Agent Communication

1. Introduction

It is over 50 years now when the first computer was invented. Since then they were regarded as a box placed on a desk in an office, lab or at home. In all cases the user knew where his computer is and needed to walk over to begin interacting with it. Only ten years ago, Mark Weiser at Xerox PARC introduced the term *ubiquitous computing* which described a vision for living environments populated with "computerized" artefacts where the emphasis was on greater user-friendliness, more efficient service support, user-empowerment, and support for human interactions [1]. That vision inspired scientists to build a new future of interaction that went beyond the desktop by computerizing everyday artefacts, which positively impacts our everyday lives. Some of this vision is already becoming a reality. Today people are increasingly surrounded by intelligent intuitive interfaces that are embedded in all kinds of artefacts. There is a need for an environment that is capable of recognizing and responding to the presence of different individuals in a seamless, non-intrusive and context sensitive way [2].

Each artefact (e.g. chair, lamp) has been designed to be suitable for certain tasks. After these artefacts become smart or even intelligent, people may have to update their task models, as they will no longer interact with a computer but with computationally enabled objects. In other words, an item can be used to activate different tasks for various services. An intelligent chair, for instance, can be used to manipulate the light level and also regulate the temperature of a room. However, the functionality and services can vary immensely depending on the purpose the human is using the chair.

Industrial and building automation institutions have been working for decades to develop network protocol standards (e.g. LonWorks and X10) supporting the issues of automatically associating intelligent artefacts. However, the automation process is more related on discovering artefacts and their services within a system. It is not possible to perform automatic or even intelligent associations between devices without the help of a user who specifically defines them.

When a new artefact enters or an existing one leaves the network (e.g. breakdown), it becomes the users responsibility to reconfigure the network accordingly otherwise the whole system would become unstable and fails to operate in the way it was designed to work. It is obvious to understand the complexity and difficulty facing humans to design a fault-tolerant system and maintain an overview in an environment with hundreds of thousands of artefacts. Thus, some form of intelligence has to work on behalf of the user to dynamically coordinate the actions of artefacts in this infrastructure. Two areas in which this is particularly evident are the system's ability to accurately determine a user's task and intention and its ability to intelligently develop associations between artefacts to assist the user in these activities.

This paper is structured as follows: The next section reviews the literature in respect to ubiquitous computing environments and agent associations. Section 3 presents the intelligent dormitory. In section 4, we introduce the intelligent associations system (IAS) with its terminologies. Section 5 illustrates initial experiments on the intelligent association system. In the last section we conclude and describe future work.

2. Related Work

The vision of ubiquitous computing environments has initiated many research projects. ETH Zurich in Switzerland, work currently on a European funded project, called SMART-Its, where their main concern is on how to get dynamically qualitative relations and selective connection between smart agents [3]. The user can take two SMART-It agents they wish to connect, move them together and establish the connection by simply shaking or waving the agents. By doing this, data

together with the SMART-Its ID is then broadcasted over a shared medium to the other agent that is within the listening range. The listening agent receives the data and establishes the connection between the two of them. This approach is easy-to-use and can be handled with a limited number of agents.

Although, the described method is currently only a prototype and not the final product it is obvious to see that this approach requires pre-knowledge on how the agents are going to be used and is too complex for the users, especially for disabled and elderly persons, to manually establish the connection between the devices by going thorough them one by one.

The HIVE project at MIT [4] is an example of a distributed agent environment. Each agent resides in a particular place (cells) and uses various local resources (shadows). The agents export selected functionality to the network and communicate with each other to share those functions. The agent interaction is ad-hoc, however the connection between the agents has to be established manually or typed. The system architecture is quite open and flexible by means of adding new agents and creating new applications. Each agent comes with a description and capabilities that are announced via broadcasting, discovered and used by other agents in the domain. This model differs from our work principally in respect that their agents are soft (rather than our hard autonomous embedded-agents) with access to hard devices via coded objects. The software agents reside on servers (e.g. PCs) and, as a consequence, do not have to consider the compactness of agent design, which is one central focus of our work.

EUNICA is a prototype of an intelligent household system developed at the Slovak University of Technology in Bratislava [5]. The user in the household is surrounded by a multitude of interconnected agent on a network that is invisible for the user. The system is intended to deliver various home-related services to the users and exhibits some kind of intelligence, e.g. it is able to recognize each individual in the household and adapt behaviour according their needs. In addition, it is able to recognize specific events (such as time or movement of a user) and act upon various situations.

The heart of the EUNICA is the control unit. Any agent can be connected to the control unit using various types of connections, such as Bluetooth, cable, etc. Monitoring and controlling devices are visualized on simple mobile Java based user interface devices connected to the control unit using the Bluetooth wireless communications technology. This can be in form of physically compact devices, such as PDAs (here "eurecos"). They display and allow browsing information received from the control unit (represented in the EUNICA Markup Language that is based on XML) and send user's requests back to the control unit.

The Scottish Centre for Environmental Design Research (SEARCH) at the Robert Gordon University in Aberdeen developed a software suite of tools called CUSTODIAN [6]. The principle aim of this research was to introduce the smart home technology to non-technical users, in particular to individuals with disabilities and elderly persons. The CUSTODIAN is a visual and flexible tool that simulates smart home networks prior to their physical installation. It can combine different existing network technologies and provides the user with a library of devices (agent). New agents can be added or removed only in the simulation. While removing an agent the tool can automatically detect which other agents need to be also removed. Although this tool is quite flexible in its nature, it is mostly used to design environments and layout the network structure for an individual person. The architects and technicians must rely on this information in order to construct the environment. There is also no way of online re-associating agents in order to get more services.

There are other high profile Intelligent Environments projects such as the Microsoft Smart House, BT's Tele-care and Cisco Internet Home [7]. However most of these industrial projects, including home automation technologies, like LonWorks and X10 (as described in section 1), are geared toward using networks and remote access with some smart control (mostly simple automation) with sparse use of AI and little emphasis on learning associations and adaptation to the user's behavior.

To the author's knowledge no work had addressed the issues related to learning associations and dynamically (re) configuring a heterogeneous set of agents within a multi-embedded agent environment.

3. Why intelligent associations?

The associations that would result the rules for the control function are either expected (defined and set by the user) or unexpected (learnt). Expected rules conform to the user's existing knowledge or expectations and the number of rules in a rule base of an agent can be calculated. However, in an environment like the iDorm, this number can reach dimensions that would make it very difficult, even impossible, for the user to specify all of them. For instance, environmental parameters of the iDorm such as light level, temperature, occupancy, humidity, chair and bed pressure sensors, bed lamp state, desk lamp state, and the blind state, heater state, cooler state would form a rule base of 3*3*2*2*2*2*2*2*2*2= 2304 possible rules. Each input is represented by a minimum number of fuzzy sets (like Temperature as LOW, MEDIUM, HIGH, and Bed and Chair Pressure as ON and OFF).

Firstly, these kind of massive numbers require a large storage space and consume a lot of the processing time during the rule search functions and performing calculation over all of these rules [8]. Secondly, these are "only" ten of hundreds possible inputs for the agent and it is obvious that the user will have considerable difficulty to manually define and analyze so many rules. Therefore, the rules have to be learnt by the agent while the user interacts with the environment. However, the learning mechanism would learn rules of input parameters that are statically defined in a rule base. In this case, all the inputs in the rule input vector is of the same importance for behaviors, although this might not always be the issue. For instance, if a user wants to switch on the desk lamp when he sits on the

chair, doesn't necessarily depend on the current light or temperature conditions of the room.

Showing the most relevant inputs in a rule for certain behaviors are more important than the whole rule itself. Learning the associations between the agents' inputs and outputs can solve this problem. Depending on this information, the rule base and its input and output parameters can be dynamically created; rules can be added, deleted and modified online by taking into consideration the importance of the inputs.

Also, during the lifetime of the system an input can become less important and an others agent's input more important for the agent. If it is of less importance (e.g. agent removal from the system or agent breakdown) the association and its related rules can be deleted and new inputs can be discovered and associated. By doing this, self-configuring, fault-tolerant multi-embedded agent systems may be created in an ad-hoc manner.

Due to the physical limitations of the agents, the bandwidth has to be kept in reasonable sizes. This means, that the agents can only be connected to a limited number agents at the same time. If an agent requires sensory information from other agents to perform its control function, it has to communicate with them to request this information. However, in a domain with hundreds of agents this would increase the communication overhead and lead to a system delay or even crash. If the agent would know only the inputs from agents that it is interested in, it would only communicate with these agents.

From the above discussion it is obvious that a user-dictated manual association system cannot deal with dynamic ubiquitous computing environments inhabited by hundreds of embedded-agents. Thus, we introduce our intelligent association system that tries to overcome the above-mentioned problems and perform intelligent associations in a non-intrusive way through the user and the environment.

4. The Intelligent Association System (IAS)

The intelligent association system terminologies and the need of learning associations between agents are described in this section.

4.1 Terminologies

A small definition of agents in a ubiquitous computing term is a physical object with an embedded processor, memory, sensors and/or actuators, and a network connection. An agent has to interact with its environment and has the ability to be able to change its state and its outputs to provide service.

We define three types of agents: (1) *Passive* agents are purely reactive and provide sensory information, (2) *Active* agents are proactive by means of executing a set of predefined rules that are stored in the computational logic, e.g. security systems, and (3) *Intelligent* agents are autonomous, adaptive and capable of learning new behaviors/rules. In this approach, we regard temperature, light level, and pressure sensors etc. as passive agents and actuators, such as lamps, blinds, heaters, and mobile robots etc. as intelligent agents.

An *Agent Society* is a collection of agents (passive and/or intelligent) that are members of a certain society. An intelligent agent within a society can become the local leader of its society, so-called *Embassador* agent (Fig. 1). A similar definition of these terms can be found in [9].

The Embassador has some specific properties and recourses and functions:

- Structure containing its members and associations between them
- Techniques of message diffusion to members of the same society to other societies
- Mechanisms of (re)-combining/dissolving associations

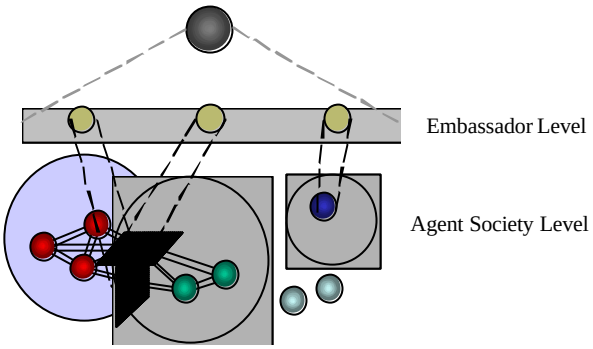

Fig. 1. Agent Societies and Embassadors

The agents within a society are associated together to form some coordinated set of agents that operate together to carry out an objective function. An association is a soft or hard link between two or more agents. The associations are established as follows: *Passive agent(s) -> Intelligent agent*. The associations can be in one of the three ways specified as follows:

- *Manual Association*, where agents are associated manually
- *Smart Associations*, where the associations are managed using an association editor
- *Intelligent Associations*, where the associations are managed by intelligence in each of the agents or the Embassador agent

4.2 The iDorm – A test bed for IAS

The Intelligent Dormitory (iDorm) forms the experimental framework for our ubiquitous computing environments research [10]. Being an intelligent dormitory it is a multi-use space (i.e. contains areas with different activities such as sleeping, working, entertainment etc) and can be compared in function to a room for elderly or disabled people or an intelligent student or hotel room. The room consists of normal furniture that will allow the user to live comfortably as it has a bed, a working desk, bedside cabinet, and wardrobe, a multi

media PC that the user can use for working or entertainment. Besides this, the room contains a large number of sensors and actuators, such as indoor and outdoor light level, temperature and humidity sensors, chair and bed pressure sensors, bed lamp, desk lamp, controllable vertical blind etc. Also the room has mobile robots that navigate within the user environment to serve the user needs such as getting drinks, medicine etc. The iDorm is populated with different kinds of embedded-agents, which can be static (e.g. controlling the building temperature), or mobile (like mobile robots) or portable (like wearable devices).

The iDorm combines three networks, LonTalk, Tini 1-wire and IP. This provides a diverse infrastructure and allows the development of network independent solutions. It also gives us an opportunity to evaluate the merits of each network [9].

The iDorm gateway server creates the standard interface for the iDorm. This exchanges XML formatted queries with the entire principal computing components, which overcomes many of the practical problems of mixing networks.

A VRML model is used to monitor and control the agent within the iDorm. The advantage of the model is that it functions alongside standard interfaces such as switches and buttons on the wall. It also allows the user to monitor and/or control the environment from a remote location.

5. Initial Experiments

5.1 The IAS Editor

The motive operating the IAS Editor is (1) it can dynamically configure and visualize associations within a multi-agent system, which does not exist in the current technology to the author's knowledge.

The association process is divided into two parts: In Part I the user associates available agents and specifies rules for them by using the IAS Editor. Part II is the actual control process of the agent where it creates its rule base and learns new associations and rules while the user interacts within the iDorm.

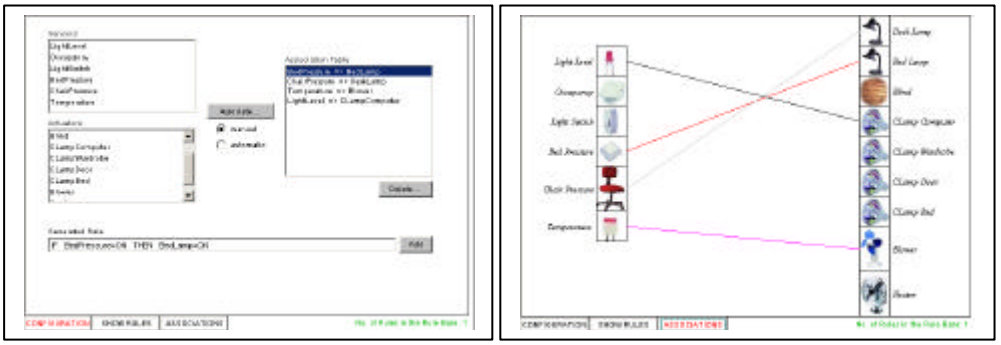

Fig. 2. (a) The Rule Generator (b) Visualization of associations

Part I: User sets associations manually or automatically with the IAS Editor

The tool is primarily concerned in assisting non-technical users during the design and set up of their living spaces. The IAS Editor is a tool focusing on representing the user's existing knowledge about associative relations on agents within the iDorm. Although it is not necessarily important to link agents and define rules beforehand, it is useful if the user's expectations from the system are known and clearly defined. Rules and associations can be set being non-changeable and removable, so that they will exist during the lifetime of the agent. This proofs to be quite important for safety rules. For instance, in case of a deaf person, the desk lamp should blink if the alarm goes on (alarm -> desk lamp).

The IAS Editor works as follows:

(1) Repeat until the user decides to stop

- The user specifies some existing knowledge or modifies the knowledge specified previously by means of associating passive agents with intelligent agents, e.g. Chair Pressure -> Desk Lamp, Temperature -> Heater, etc.
- The user enters the rules for the established associations with the help of the rule generator (Fig. 2 (a)), e.g. IF Chair Pressure=ON THEN Desk Lamp=ON, IF Temperature=LOW THEN Heater=ON, etc.

- The user saves the rules and/or removes existing rules

One of the main features of the IAS Editor is the visualization panel that dynamically draws the associations that has been set by the user (Fig. 2 (b)). In the rule visualization panel the IAS Editor depicts the entered expected rules also entered by the user.

(2) Send generated initial rule base to the agents

- The IAS Editor connects to the iDorm agent and sends the rule base via socket connection
- The message being sent to the agent is in the form of a XML message as show below:

```
<IAS>
    <NORULES>2</NORULES>
    <SENSORS>CPRESSURE; TEMPERATURE </SENSORS>
    <ACTUATORS>DESKLAMP; HEATER</ACTUATORS>
     <RULE0>
          <IF>CPRESSURE=ON</IF>
          <THEN>DESKLAMP=ON</THEN>
   </RULE0>
   <RULE1>
          <IF>TEMPERATURE=LOW</IF>
          <THEN>HEATER=ON</THEN>
   </RULE1>
</IAS>
```

Part II: Rule base generation and agent control/learning process

After receiving the XML message the agent starts parsing the message and forming the associations with the chosen agents. In this case the rule base of this specific agent consists of two inputs (CPRESSURE, TEMPERATURE) and two output parameters (DESKLAMP, HEATER). Then, the rules are entered into the newly created rule base. Table 1 below shows the rule base of the agent with the rules specified by the user:

Table 1. Rule base, specified by the user

CPRESSURE	TEMPERATURE	DESKLAMP	HEATER
ON		ON	
	LOW		ON

It is noted here, that the associations and rules are learnt simultaneously in a dynamic manner as follows:

As soon as the rule base has been generated the agent goes into the monitoring mode. During the monitoring mode, the agent monitors the states of every agent that is currently available within the iDorm. As long as the state of the intelligent agents (desk lamp, heater, cooler, ceiling lamps etc) doesn't change, the monitoring mode stays active. As soon as the user changes the state of an actuator (e.g. by switching on the bed lamp) the agent takes a snapshot of the current environment state and switches to the learning mode. The learning mechanism is based on instance-based learning. The current environment states of the agents are compared with the previous states and the passive agents that have changed states get associated with the agent that caused the change. For example, the user switched on the bed lamp because he sits on the bed. (1) The Bed Pressure sensor changed the state from OFF to ON. (2) And also the desk lamp caused the light level sensor to increase by 30%. This means that the bed lamp state change is associated to the bed pressure and the light level sensor.

Bed Pressure -> Bed Lamp
Light Level -> Bed Lamp

In order to avoid wrong association settings because of sensory noise, all the sensors monitored go through a filter. The filter is currently set to 10%. Any change above this filter may have resulted from the agent's state change.

Now the rules and the new interested passive agents are entered into the rule base. Table 2 below shows the rule base of the agent after the agent has learnt the necessary association. Note that the IAS had inserted two new associations and their corresponding rule by learning from the human actions.

Table 2. Rule base, after learning new associations[1]

CP	BP	TP	LL	DL	HT	BL
ON				ON		
		LOW			ON	
	ON		LOW			ON

After updating the rule the agent sends a XML message with the updated associations and rules back to the IAS Editor that illustrates the changes on the visualization panel.

5.2 Inter-agent communication

We are currently involved in a collaborative project with the Korea Advanced Institute of Science and Technology (KAIST) supported by UK-Korea S&T collaboration fund. The project is concerned with tele-monitoring and tele-caring elderly and disabled persons over long distances. Here, our UESSEX iDorm-Agent in the UK is associated with the KAIST robot agents (RA) in Korea over the Internet. This association enables the iDorm-Agent to use the services provided by the KAIST RA, such as delivering newspaper and medicine to a bed-bounded person. In response, the RA uses services to change the environmental settings of the iDorm provided by the iDorm-Agent. For instance, the iDorm-Agent commands the RA to deliver the medicine for the patient. After successfully finishing the task, the RA may send a notification command back to the iDorm-Agent to switch on the desk lamp to which the iDorm-Agent is associated with. It should be noted that in this experiment the associations are set manually since the main focus is on agent interaction and communication.

The current interaction and communication

[1] Abbreviations:
CP = Chair Pressure, BP = Bed Pressure, TP = Temperature
LL = Light Level, DL = Desk Lamp, HT = Heater, BL = Bed Lamp

mechanism is designed for heterogeneous agents: when two or more agents are associated and exchange messages, the messages should be known and have the same semantics for all of them. The agents should share the same languages, protocols and ontologies. We have developed DIBAL, the Distributed Intelligent Building Agent Language for embedded agents. The benefit of this messaging framework in comparison to other existing agent communication languages, such as KQML and FIPA mostly lies in its restricted set of primitives that reduces the functional complexity and required memory size drastically, and compact, tagged and hierarchical structure. The following introduces the messaging framework. A more exhaustive description on DIBAL can be found in [11].

DIBAL defines three main primitives for the messages: *send*, *receive* and *notify* (Fig. 3). The *send* message is used to send commands to the remote agent, the *request* message to receive agent state and location information and the *notify* message to sends either feedback on the request or any "abnormal" situations/behaviors of the agent. Abnormal behaviors can be regarded as broken or not available.

The first tag of the message consists of the domain where the message should be sent. This includes the UESSEX iDorm-Agent, and the KAIST Intelligent Sweet Room (ISR) in general, and MANUS Arm, Pioneer II mobile robot, the vision system, in particular.

Tag 2 defines tasks and/or control details of each agent. A task can consists of a sequence of predefined sub-tasks. For instance, the KAIST RA can have a predefined behavior that picks up medicine and delivers it to the bed-bounded person.

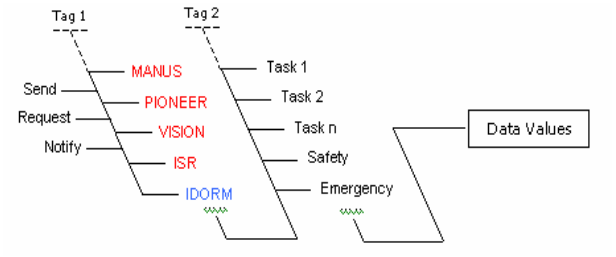

Fig. 3. The Message format for the common communication protocol

The network communication, based on stream sockets, between the iDorm-Agent and the RA is established by initiating a request command from the iDorm-Agent to the RA server. The server creates the link between the agents and waits for incoming messages/commands from the iDorm-Agent. Once the command has been sent the server passes the request to the responsible RA to fulfill the task and informs the iDorm-Agent about the current status of the robots. If the task is not complete then the server sends a message indicating that the job is "INCOMPLETE". Every time when the iDorm-Agent wants to send out a new command, it waits until the previously requested job has been successfully finished. The issue of time delay and packet loss is not the main subject in this project but it has been addressed by applying a feedback system.

6. Conclusions and Future Work

Identifying the associations between agents in a multi-agent system provides a way of understanding the nature of the system, its purpose and functionality. This is particularly important for a system with a large number of agents and is highly dynamic in nature. It is very difficult, even impossible, for a non-technical user to maintain all of the existing agents and establish connections among them.

In this paper we have presented an intelligent association system (IAS) applied in ubiquitous computing environments. The system we have described is divided into two parts, (1) manually associating agents and specifying their rules with by using the IAS Editor. (2) Dynamically creating agents' rule bases and learning rules and associations. This all happens online while the agent interacts with the environment, including its occupant.

We have also argued that learning association can bring significant benefits, like:

- Reduces storage and processing capacity, and the communication overhead
- Overcomes the dimensionality problem in rule bases through extracting/deleting rules and modifying the input-output vector
- No requirement of pre-knowledge of the agents existing within the society
- No requirement on expert knowledge for setting up the association, generating the rule base and entering the rules
- Ad-hoc, Fault-tolerant, and self-configuring agent societies without the need of an network operator

The paper also documented on the experimentations undertaken. The first experiment is concerned on manually creating and learning associations. The second experiment conducted link agents in physically separated intelligent spaces together (here UK to Korea) in order to tele-care and tele-monitor elderly people and/or persons with disabilities. This experiments main purpose was to create a common messaging framework.

In the future we will investigate the degree of importance (association weights) of associations. In the current version of the IAS the associations are

either established or not. Through the lifetime of the system an agent might get less important to an agent that it is associated with. We believe that rules and the associations should be grouped according to their degree of importance. This lets the intelligent agent to do better decision on which rules it needs to execute in case of conflicting situations. In addition, in the future we plan to significantly expand the agent set and explore more fine-grained and course grained distributed embedded -agents.

Acknowledgement

We are pleased to acknowledge the funding support from the EU IST Disappearing Computer program (eGadgets) and the joint UK -Korean Scientific fund (careAgents), which has enabled much of this research to be undertaken. We especially acknowledge the work and effort from our Korean partners from KAIST who are co-operating with us in the Robotic and Building agent communication. In particular we would like to acknowledge the highly valuable advise from Professor Zenn Bien of KAIST and his team, which includes Mr. Kim and Mr. Lee and Mr. Myung.

We would also like to thank Anthony Pounds-Cornish, Arran Holmes, and Filiz Cayci for their indirect contributions arising from many stimulating discussions on intelligent artefacts and embedded agent issues.

References

[1] M. Weiser, "The Computer of the 21st Century", *Scientific American*, Vol. 265, No. 3, pp. 66-75, September 1991.

[2] V. Callaghan, G. Clarke, M. Colley, H. Hagras, "Embedding Intelligence: Research Issues for Ubiquitous Computing", *International Conference on Ubiquitous Computing in Domestic Environments*, Nottingham, UK, September 2001.

[3] L. E. Holmquis, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, H. W Gellersen, "Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts", *Ubicomp 2001*, Springer-Verlag LNCS 2201, pp. 116-122, 2001.

[4] N. Minar, M. Gray, O. Roup, R. Krikorian and P. Maes, "HIVE: Distributed Agents for Networking Things", *ASA/MA*, 1999.

[5] M. Bielikova and T. Krajcovic, "Ambient Intelligence within a Home Environment", *ERCIM News*, No. 47, pp. 12-13, October 1991.

[6] J M. Martins Ferreira, T. Amaral, D. Santos, A. Agiannidis, "The Custodian Tool: Simple Design of Home Automation Systems for People with Special Needs", EIB Scientific Conference, Munich, October 2000.

[7] A. Sherwin, "Internet home offers a life of virtual luxury", *The Times magazine*, pp. 10, 3rd November 1999.

[8] H. Hagras, V. Callaghan, M. Colley, G. Clarke, H. Duman, "Online Learning and Adaptation for Intelligent Embedded Agents Operating in Domestic Environments", *Fusion of Soft Computing and Hard Computing for Autonomous Robotic Systems, Studies in Fuzziness and Soft Computing Series*, Physica-Verlag, 2002.

[9] N. Hanssens, A. Kulkarni, R. Tuchida, T. Horton, "Building Agent-based Intelligent Workspaces", *International Workshop on Agents for Business Applications*, Las Vegas, NV, 2002.

[10] A. Holmes, H. Duman, A. Pounds-Cornish, "The iDorm: Gateway to Heterogeneous Networking Environments", *International ITEA Workshop on Virtual Home Environments*, Paderborn, Germany, February 2002.

[11] F. Cayci, V. Callaghan, G. Clarke, "A Distributed Intelligent Building Agent Language (DIBAL)", *SCI/ISAS 2000*, Orlando, Florida, July 2000.

