Soft Computing Agents
V. Loia (Ed.)
IOS Press, 2002

In the book entitled "Soft Computing Agents: A New Perspective for Dynamic Information Systems", in the International Series "Frontiers in Artificial Intelligence and Application", (Ed: V. Loia), IOS Press, pp.25-54, December 2002

Chapter 2

Incremental Synchronous Learning for Embedded Agents Operating in Ubiquitous Computing Environments

Hani Hagras
Victor Callaghan
Graham Clarke
Martin Colley
Anthony Pounds-Cornish
Arran Holmes
Hakan Duman

2.1 Introduction

In this chapter we introduce a novel learning and adaptation mechanism for embedded-agents that are embodied in devices making up ubiquitous computing environments. We illustrate the concept using eGadgets, which are types of ubiquitous computing devices developed by ourselves and our European partners, CTI (Patras, Greece) and NMRC (Cork, Ireland), as part of the EU Disappearing Computer programme. The concept of eGadgets is to create a conceptual and technological framework that will allow ordinary people to assemble and use, with ease, a collection of network-aware computer based products to provide collective functionality that will empower their lives beyond that provided by today's stand-alone products. Integrating useful amounts of intelligence into embedded devices is an essential enabling technology to achieve the vision of ubiquitous computing. In support of this vision we describe a fuzzy logic based Incremental Synchronous Learning (ISL) technique that provides online life-long learning that can be operated in a non-intrusive mode. A unique feature of our learning mechanism is that it seeks to *particularise* the agent's learnt behaviour to the individual user rather than work on behalf of the machine or adjust itself to the average behaviour of users. In order to assess such technology we have constructed the iDorm (intelligent dormitory) to act as a test-bed for intelligent inhabited environments. We report on an initial experiment in this new facility in which our intelligent embedded agents, powered by our ISL learning mechanism, learn a user's behaviour and control the iDorm for two days.

2.2 Project Framework

"Startrek" and similar science fiction films and series paint an intriguing picture of the future, one in which masses of unseen and tireless electronic devices and intelligent-agents attend to occupants every need; regulating the air they breath, the temperature of their cabins, their entertainment and communications. In fact their very existence in such alien environments is wholly dependent on technology. For many, space exploration and planetary habitats are not just the "final frontier" for mankind but the ultimate vision for ubiquitous computing and ambient intelligence. Fuelled by advances in microelectronics and Internet technology the variety of current computer-based networked artefacts, is

growing at a huge rate. Some recent estimates showing of the order of 8 billion microprocessors were produced in 2001, of which only 2% went into PCs, the rest going into embedded computer devices most people wouldn't recognize as computers (Callaghan *et al*. 2000b). Such devices vary from, mobile telephones through home entertainment systems to cars. Thanks to pervasive networking (e.g. the Internet) such machines and artefacts can be configured by users into personalised and novel arrangements so that devices are able to coordinate their actions to support peoples lives. An obvious barrier to achieving this vision is the inherent complexity of the technology. People must be able to use such devices without needing to understand or deal with the underlying technology. One approach to achieving this is to embed intelligent agents into the devices that make up ubiquitous environments thereby transferring some of the cognitive load from people to machines; the extent to which this is done being sometimes referred to as the "cognitive disappearance metric".

It is anticipated that all forms of goods will be influenced by this development from items that are clearly electronic in nature today (e.g. mobile phones, home entertainment systems, kitchen appliances, etc.) to those that are currently not (e.g. clothing, desks, etc.) (Callaghan *et al.* 2000a). Such goods will find themselves in a variety of Intelligent Inhabited Environments (IIE); spaces such as cars, shopping malls, homes, clothes and even within our own bodies. However, in order to realise this vision, technologies must be developed that will support ad-hoc and highly dynamic (re)structuring of such networked arrangements of embedded computing devices whilst, wherever possible, shielding non-technical users from the need to understand or work directly with the underlying technology. The authors are engaged along with our European partners CTI and NMRC in the eGadgets project [http://www.extrovert-gadgets.net/] funded by the EU "Disappearing Computer" programme. A programme which, in part, aims at the development of compact intelligent embedded-agents (intelligence integrated into computational artefacts) and computational architectures to assist with the above.

The work proposed by this chapter focuses on the investigation and development of learning and adaptation techniques that seek to provide an online, life-long, personalised learning of anticipatory adaptive control in devices making up ubiquitous computing environments exemplified by our EU eGadgets project. To these ends we introduce our Fuzzy Incremental Synchronous Learning techniques and describe our intelligent dormitory (iDorm) as a testbed for our work in learning and adaptation, together with supporting experiments and results.

2.3 eGadgets

An eGadget is a tangible object (which can be an everyday object), it has a communication module (wired, radio or infrared), it has at least one plug (the abstraction of the ability to co-operate with other eGadgets). An eGadget also has a digital self (software running on the eGadget, on a host computer or both) and may or may not have processing power and memory and may or may not have sensors and actuators and may or may not be able to learn to adapt its operation to meet the users needs.

The concept of eGadgets is to create a conceptual and technological framework that will allow ordinary people to assemble and use, with ease, a collection of network aware computer based products (e.g. cellphones, music-players, washing machines, heating systems etc) to provide collective functionality which will empower their lives beyond that possible by today's stand-alone products. Embedding useful amounts of intelligence into eGadgets environments is an essential enabling technology to achieve the vision of the eGadgets (Callaghan *et al.* 2000b). In the terminology adopted by this project, individual devices are called eGadgets; collections of connected eGadgets are called GadgetWorlds;

the conceptual and technological framework is referred to as Gadgetware Architectural Style (GAS).
An example of a GadgetWorld might be a room in an intelligent-building in which environment and entertainment based eGadgets work together to form an integrated interactive environment. In this chapter we will describe our novel methods that supply compact reasoning and learning mechanisms that are able to deal with the numerous inputs and highly dynamic environments (both in the physical and network sense) that will characterise GadgetWorlds and provide useful amounts of intelligent functionality. We will use the iDorm as an example of GadgetWorlds and will show how can our Incremental Synchronous Learning (ISL) techniques provide an online, life-long, user-particularised, anticipatory, adaptive, control agent for a intelligent inhabited environments.

2.4 Intelligent Embedded Agents operating in Intelligent Inhabited Environments (IIE)

Ideally, for the vision described in the introduction to be realised, people must be able to use computer-based artefacts (or eGadgets) and systems without being cognitively aware of the existence of the computer within the machine. Clearly in many computer based products the computer remains very evident, for example, with a video recorder, the user is forced to refer to complicated manuals and to use his own reasoning and learning processes to use the machine successfully. This situation is likely to get much worse as the number, varieties and uses of computer based artefacts increase. We argue that if some part of the reasoning, planning and learning normally provided by artefact user, were embedded into the artefact itself, then, by that degree, the cognitive loading on the user would reduce and, in the extreme, disappear (i.e. a substantial part of the computer's presence would disappear). Put another way, the proportion of reasoning, planning and learning transferred to the artefact or eGadget (collectively referred to as "embedded-intelligence") is a measure of cognitive disappearance. Hence we view embedded intelligence as an essential property of artefacts for the cognitive disappearance of the computer and necessary to the successful deployment of new technology in Intelligent Inhabited Environments (IIE).
Embedded-computers that contain such embedded intelligence are normally referred to as "*embedded-agents*" (Callaghan *et al.* 2001). It is now common for such embedded-agents to have an Internet connection thereby facilitating multi embedded-agent systems. In a fully distributed multi embedded-agent systems each agent is an autonomous entity co-operating by means of either structured or ad-hoc associations with its neighbours.

2.4.1 Other work related to development of Intelligent Embedded Agents for IIE

There are a growing number of research projects concerned with applying Artificial Intelligence (AI) and intelligent agents to IIE. In Sweden, Davidsson (Davidsson 1998) utilises multi-agent principles to control building services. These agents are based on the AI thread that decomposes systems by function rather than behaviour as in our research. Their work does not address issues such as occupant based learning. In Colorado Mozer (Mozer 1998) uses a soft computing approach - neural networks - focusing solely on the intelligent control of lighting within a building. Their system, implemented in a building with a real occupant, achieved a significant energy reduction, although this was sometimes at the expense of the occupant's comfort. Work at MIT on the HAL project (Brooks 1997) concentrates on making the room responsive to the occupant by adding intelligent sensors to the user interface. In the University of Loughborough Angelov (Angelov *et al.* 2000)

looked at the application of fuzzy rule-based models in HVAC system simulation and he was concerned with producing optimal models for buildings, which could be used later in control. The HIVE project at MIT (Minar *et al.* 1999) is an example of a particularly forward-looking distributed agent model. This model differs from our work principally in that their agents are soft (rather than our hard embedded-agents) with access to hard devices being via coded objects referred to as shadows. The soft 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. The University of Reading is active in the field of intelligent buildings and their view of intelligence is rooted in structural design and building utilisation concepts. Currently their research is mostly focusing on monitoring people over the network with the "talking sign" chips. There are also other high profile Intelligent Building projects such as the Microsoft Smart House, BT's Telecare and Cisco Internet Home. However most of these industrial projects 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 and adaptation to the user's behaviour. To the author's knowledge no other work has addressed the online learning and adaptation of intelligent embedded agents to specific users habitual behaviour operating within IIE.

2.4.2 Some Challenges facing Research in Intelligent Embedded Agents

Traditional AI techniques are well known for being computationally demanding and therefore unsuitable for 'lean' computer architectures. Historically most traditional AI system were developed to run on powerful computers such as workstations, whose specifications are at least two orders of magnitude removed from most embedded-computers. In addition traditional AI techniques have proved too fragile to operate real time intelligent machines. As a result, even implementing simplified traditional AI systems on embedded-computers has proved a considerable challenge to computer science.
Another problem is that most automation systems (which involve a minimum of intelligence) utilise mechanisms that generalise actions (e.g. set temperature or volume that is the average of many people's needs). However, we contend that AI applied to intelligent environments needs to particularise itself to the individual. A key underlying concept in the eGadgets model being that the technology should empower people to freely design novel configurations (often, not envisaged by the original gadget designers) thus providing a form of emergence. A fundamental axiom of our approach is that "the user is king", the users intentions (in the form of their usage actions) should, wherever possible (safety being one exception), be faithfully reflected in the eGadget or GadgetWorlds operation. In other words, programming or learning should support the notion of particularisation over generalisation, which is more common in other application domains. Thus, the value of an intelligent embedded agent lies in the agent's ability to learn and predict the human and the system needs and automatically adjust the agent controller to meet them and the agent's ability to do such learning and prediction based on a wide set of parameters. There is thus a need to modify effectors for environmental variables like heat and light etc on the basis of a *complex multi dimensional input vector*, which cannot be specified in advance. For example, something happening to one system (e.g. reducing light level) may cause a person to change behaviour (e.g. sit down) which in turn may result in them effecting other systems (e.g. needing more heat). An agent that only looks at heat levels is unable to take these wider issues into account. An added control difficulty is that people are essentiality non-deterministic and highly individual, therefore as explained above there is a need for a system that particularises for individual users rather than generalising for a group of users. When viewed in such integrated control terms it is possible to see why simple PID or fuzzy

controllers are unable to deal satisfactorily with the problem of online learning for embedded agents.
The quality of agent decisions is limited by its knowledge of the world. It gets its knowledge from sensors directly attached to it and other agents (i.e. indirectly from their sensors). Naturally the question arises, which set of sensor information is sufficient for an agent to make a particular class of decision? Consider a simple heating controller and ask the question, "Why does the room's occupant alter the heat value?" Is it to do with the current temperature, the current level of activity of the user, what the user is wearing, where the user is in a room, where the user has just been or what? We may decide that it is based upon current temperature and therefore could operate with only one sensor, but later discover that an agent that used only one sensor was not working very effectively. At the other extreme we could decide we should sense 'everything' and then let the agent learn which of these sensed values was important. Clearly in this latter situation the agent would be able to make better-informed decisions and adapt to changing criteria. In addition this problem exposes a central dilemma, what is the best mechanism for selecting relevant sensory sets for agents? Is it the designer or the agents themselves? The problem with a designer is the assumption that people know best what the intelligent agent needs; but is this true (a dilemma sometimes referred to as the perception gap (Callaghan *et al.* 2000a)? We would argue that it is better to provide a large set of sensory inputs to agents and let them resolve which of the stimuli is important for any given decision wherever possible. Whilst this latter argument may have some appeal it carries with it a penalty, the need to compute using large sensory input vectors. Thus, large sensory sets are an issue for ubiquitous computing environments. One solution would be the development of mechanisms to allow embedded-agents to "focus" on sub-sets of data relating to specific decisions or circumstances.

2.5 The iDorm — A Testbed for Ubiquitous Computing and Ambient Intelligence

We have chosen the Essex Intelligent Dormitory Pictured in Figure 1 to be a demonstrator and test-bed for our intelligent learning and adaptation techniques applied to eGadgets and GadgetWorlds. Being an intelligent dormitory it is a multi-use space (i.e. contains areas with differing activities such as sleeping, working, entertaining etc) and can be compared in function to a room for elderly or disabled people or an intelligent student or hotel room. The room looks like any other with normal furniture that will allow the user to live comfortably as it has a bed, a working desk, bed side cabinet, wardrobe, a multi media PC which the user can use for working or entertainment as it has the capability of audio entertainment via playing music CD, radios using an up to date Dolby sound systems and it can also display normal TV programs and DVDs. The layout of the iDorm is shown in Figure 2.
In order to make the iDorm as sensitive as we can to the needs of the occupant we need to be able to comprehensively monitor activity in the room. For these reasons the iDorm is equipped with an array of embedded sensors such as temperature, occupancy, humidity and light level sensors, as well as a camera to be able to monitor what goes inside. It is possible to follow the activities inside the iDorm, via a live video link over the Internet (Pounds-Cornish and Holmes 2002) though this is not directly involved in our attempt to develop embedded intelligent mechanisms.
The iDorm makes provision for control of numerous systems such as entertainment, office-work and environmental control. In building the iDorm, the commercial reality is that the devices we have installed reside on several different types of network so that access needs

to be managed and gateways *can therefore be regarded as critical components in such systems*, combining appropriate granularity with security. Currently the iDorm is based around three networks to be described later which are Lonworks, 1-wire (TINI) and IP although it would be possible to include others, providing a diverse infrastructure and for the development of network independent solutions (Homes *et al.* 2002). In what follows we summarise the sensors and actuators used.

Figure 1. Photo of iDorm

The room has eleven environmental parameters to be measured, which are;

- Time of the day (I1) measured by a clock connected to the 1- wire network
- Inside room light level (I2) measured by indoor light sensor connected to the Lonworks network
- Outside outdoor lighting level (I3) measured by an external weather station connected to the 1-wire network
- Inside room temperature (I4) measured by redundant sensors connected to the Lonworks and the 1-Wire networks
- Outside outdoor room temperature (I5) measured by external weather station connected the 1- wire network
- Whether the user is using his audio entertainment (I6) on the computer – either the radio or the CD player are sensed by Visual C++ code when running the Winamp program
- Whether the user is lying or sitting on the bed or not (I7) measured by a pressure pad connected to the 1-wire network
- Whether the user is sitting on the desk chair or not (I8) measured by a pressure pad connected to the 1- wire network
- Whether the window is opened or closed (I9) measured by a reed switch connected to the Tini-1 Wire network
- Whether the user is working or not (I10) sensed by a Visual C++ code that senses if the user is working on a Word document
- Whether the user is using video entertainment (I11) on the computer - either a TV program (via WinTV) or a DVD using the Winamp program sensed using Visual C++ code.

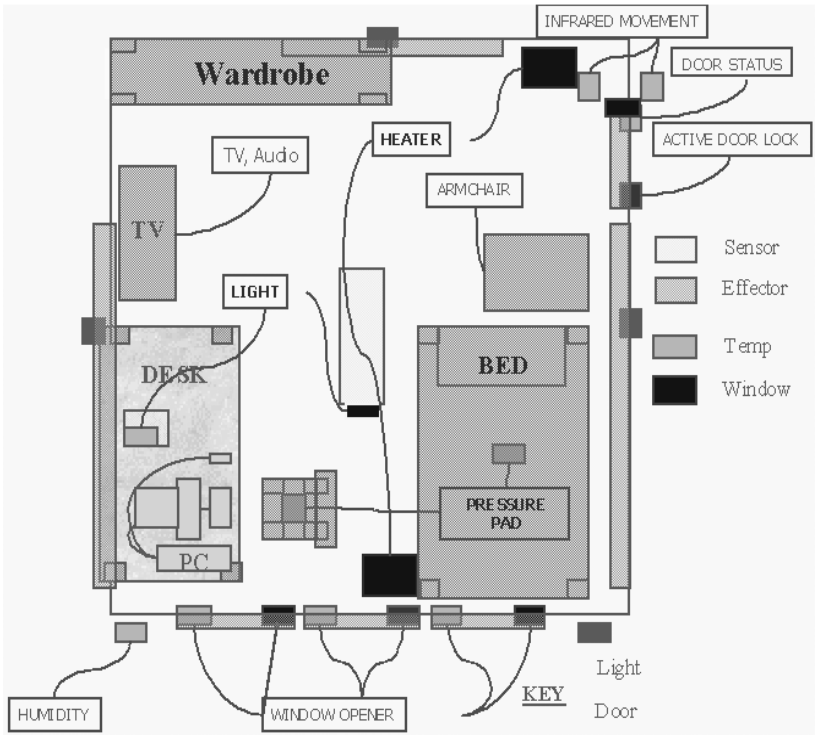
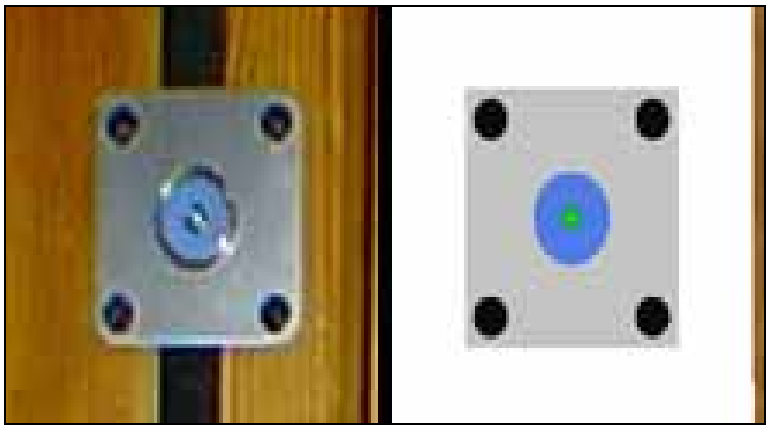


Figure 2. Layout of iDorm.

There are ten outputs to control;

- Fan Heater (O1)
- Fan Cooler (O2)
- A dimmable spot light above the Door (O3)
- A dimmable spot light above the Wardrobe (O4)
- A dimmable spot light above the Computer (O5)
- A dimmable spot light above the Bed (O6)
- A Desk Lamp (O7)
- A Bedside Lamp (O8)
- Whether the automatic blinds are opened or closed (O9)
- If the automatic blinds are closed their opening can be controlled (O10)



(a) (b)

Figure 3. The iDorm Lock a) Real World b) VRML Model.

The room is also supplied by other sensors like smoke detector, humidity sensor, activity sensors, telephone sensor to sense whether the phone is on hook or off hook. However the above set of sensors and actuators are the ones used by our learning system
The Agent is designed to learn behaviour related to different individuals. In order to achieve this, it needs to be able to distinguish between users of the environment. This is achieved by using an active lock, designed and built by our research team based on Dallas Semiconductors 1-Wire protocol.
Each user of the environment is given an electronic key, about the size of a penny. It is mounted onto a key fob and contains a unique identification number inside its 2-kilobyte memory. The door to the iDorm contains an electronic keyhole that reads the address of any electronic key held against it; the lock is depicted both in the real world and in the VRML GUI in Figure 3.
The address is dynamically compared to a small user database where the address of a key is the index to the list of information shown in Table 1.

Table 1. Information held about each Key Holder

Last name
First name
Time last entered iDorm
Time last exited iDorm
Status (Student, Staff or Guest)
Unique ID Number
Electronic Key Unique ID Number

The Unique ID Number of the user is passed to the embedded agent so that it may retrieve and update rules learnt about that user previously. The lock is also able to support multiple connections to interrogate its state.
By gathering information from its sensors over a period of time, the embedded agent can notice how a particular person tends to react to particular circumstances, and can then *learn* to replicate that behaviour itself. Using our active lock system we can distinguish between different users, the system is able to learn different behaviours for different people. So for example, the agent might learn that Person A, who is only partially sighted, prefers a higher level of light than Person B, whose sight is normal. It could then adjust the lighting level appropriately, according to who was using the room at that time.

2.5.1 The iDorm Networking

Technology for networking building services is already widely deployed (e.g. Lonworks, BACnet, EIBUs, etc), as is technology for connecting domestic and mobile appliances (e.g. Cebus, Bluetooth). These are opening up the opportunities and technical difficulties afforded by highly connected and dynamic embedded-computing and networked gadgets.
One way intelligent embedded-agents are often deployed in the creation of intelligent environments (Sharples *et al.* 1999) is to assign an agent to control a localised space (e.g. a room, a body). These spaces can map quite naturally onto to the domain managed by network gateways. We believe that a marriage of network topologies and agent architectures would hold many synergetic advantages such as (Holmes *et al.* 2002):

1. The gateway domain could be readily made to match that of an intelligent building agent (i.e. both human activity and agent control equate to spaces such buildings, floors, rooms and bodies).
2. The gateway can bridge diverse sets of data and control networks (e.g. IP and LonTalk)
3. The gateway can provide a means of managing secure access.

4. The gateway can act as an area wide server providing managed access to the resources within its jurisdiction (e.g. remote home control via a web interface).
5. The gateway can "mine" information on activity under its control (for example, information on occupant's behaviour and use of equipment within the environment might be gathered and made available for the benefit of the occupant. In this case there would have to be agreement between individuals and manufacturers say on equipment usage.
6. The gateway could provide a low cost way of providing computational resource for agent deployment (i.e. gateways will be there anyway).
7. The gateway concept is both scalable and extensible in a downward (micro-world), upward (macro-world) and horizontal (more of same) direction.

For instance, in an intelligent building (which is an example of intelligent inhabited environments) there could be a gateway at the house level, further gateways to each room and gateways to complex devices within rooms (e.g. hi-fi systems) or body-wearable devices (e.g. with the mobile phone being the gateway). The use of gateways is therefore a scaleable solution to the problem of giving a degree of autonomy and security to different levels of any complex and hierarchical system (Holmes *et al.* 2002). In our example - the iDorm - there are in fact a number of sub-networks that the common interface disguises and protects at the same time.

The iDorm uses three main communication protocols to allow its devices to communicate with each other. Such a variety of networks and protocols were chosen because any successful intelligent agent produced for the iDorm can be shown to be network independent. In the following subsection we discuss the three main networks and their communication protocols.

2.5.1.1 Lonworks Network

Lonworks is Echelon's proprietary network and encompasses a protocol for buildings automation. It is a twisted pair network, similar to IP that comes in two flavours – one that provides power to the devices through the network and another that requires devices to have an external power supply. There are many commercially available sensors and actuators for this system. Each device has typed inputs and outputs. The Lonworks system allows association to be set up between inputs and outputs using a standard PC that is connected to the Lonworks network. The PC can then be disconnected and the associations will continue to function. The system has no central coordination system, just a set of devices. The physical network installed in the iDorm is Lonworks TP/FP10 network. The gateway to the IP network is provided by Echelon's iLon 1000 web server. This allows the states and values of sensors and actuators to be read or altered via a standard web browser using HTML forms. The majority of the sensors and effectors inside the iDorm are connected via a Lonworks network as shown in Figure 4.

2.5.1.2 Wire Network

Dallas semiconductor developed the 1-wire network protocol. It was designed for simple devices to be connected over short distances. 1-wire offers a large range of commercial devices including small temperature sensors, weather stations, ID buttons and switches. Unlike Lonworks the 1-wire system has a central coordination system. The 1-wire network is connected to a Tiny Internet Interface board (TINI board) which runs an embedded Java Virtual Machine (JVM). In the iDorm the Tini connected to the 1-wire network runs an embedded web server that serves out the status of the networked devices using a Java servlet. The servlet collects data from the devices on the network and responds to HTTP requests. The network layout and attached sensors are shown in Figure 5.

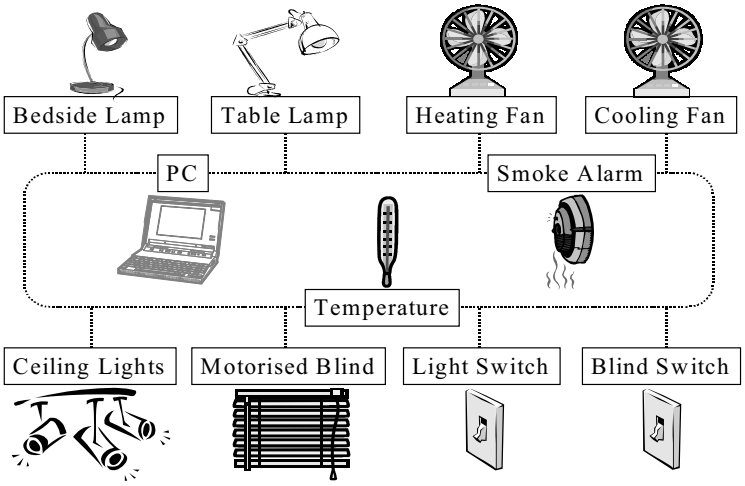


Figure 4. LonTalk network layout.

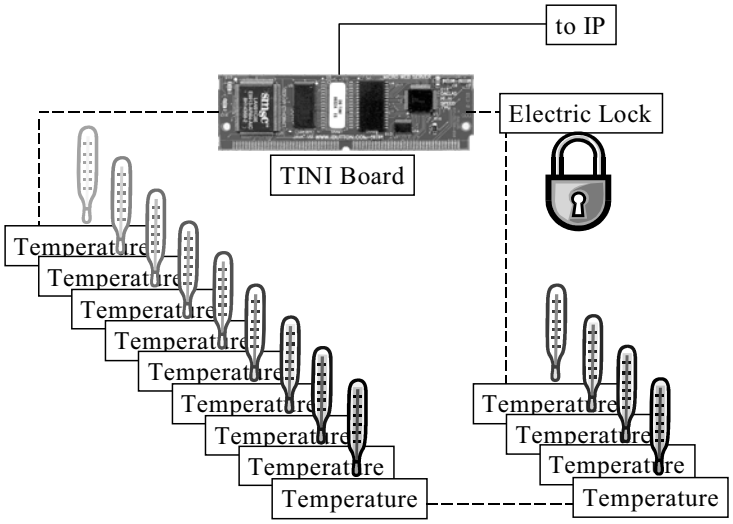


Figure 5. 1-Wire network layout

2.5.1.3 IP Network

The IP network forms a backbone to interconnect all networks and other devices like the Multi-media PC (MMPC). The MMPC will be the main focus for office-work and entertainment in the iDorm. Again the MMPC uses the HTTP protocol to display its information as a web page.

2.5.2 iDorm Gateway Server

The iDorm uses a single network (IPv4) as shown in Figure 6 to link the different networks together (Homes *et al.* 2002). This allows a common protocol to be produced so that all interfaces could use to communicate with the iDorm. There are several distinct advantages to this approach:

- The first is that a common interface immediately creates a scalable environment. More sensors can be added to existing networks or entirely new network protocols can be added to the iDorm without having to re-configure every other network that communicates in the room.
- The second is robustness. More than one network can provide similar information, if one fails the other can seamlessly provide that information. For example, the iDorm has temperature information available on both the Lonworks network and the 1-Wire network.
- The third advantage is that a common interface doesn't limit an interface to a certain way of expressing data. If all the iDorm's environmental information is available as simple states and values then it is entirely up to the interface designer as to how and in what format that data is used.
- The fourth advantage is that of security. If the iDorm's information is available through a single communication protocol, it is far easier to decide whether the client is entitled to receive this information. This entitlement can be decided on anything from identification or time. We use the latter concept to timeshare access to the iDorm when more than one experiment needs to run at one time.
- The fifth advantage is that a common protocol allows a dynamic interface to be created. An example of this is the voice recognition interface explained later in this chapter.
- The sixth advantage is that the processing power required to gather information from the room is greatly reduced by placing the onus on the common protocol to provide the information. This system reduces the amount of processing required from the interface.

The protocol that has been produced is an XML definition for the iDorm. All information requested from the iDorm must go through a central server. This server communicates with the iDorm's LonTalk and 1-Wire network across IP using HTTP requests to get environmental information and request changes to the states of the effectors

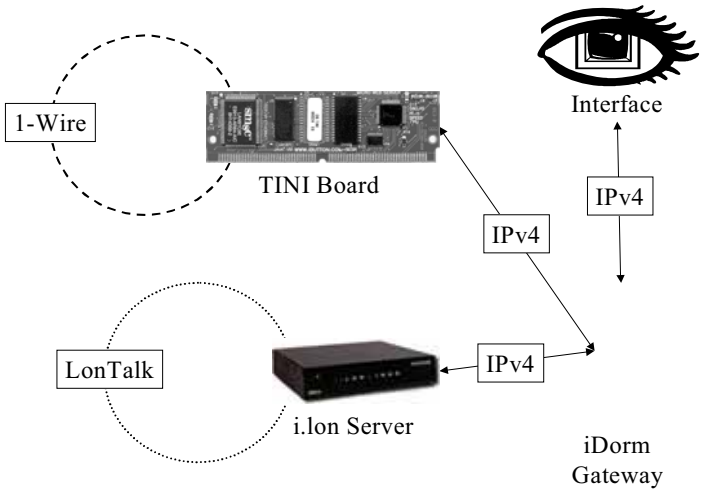


Figure 6. The high-level network

To create a standard interface to the iDorm we have an "iDorm gateway server". This receives XML formatted queries from an agent, then connects to all the network interface servers in turn and sends an HTTP GET request to make any changes to the network devices requested by the XML request. It then receives an HTML formatted page describing the states of all the devices on each network. This document is then parsed to remove the correct information and formatted in XML to retain its context, before being

returned to the requesting agent as shown in Figure 7. XML is very portable and easy to parse which makes it ideal as cross platform communications syntax. Its other main advantage is that it is human readable and can be displayed easily in a number of ways depending on the applied style sheet.
The iDorm's gateway server is a practical implementation of an HTTP server acting as a gateway to each of the room's sub networks. This illustrates the concept that using a hierarchy of gateways it would be possible to create a scaleable architecture across such heterogeneous networks in IIE (Homes *et al.* 2002). The iDorm gateway server allows a standard interface to all of the room's sub networks. There could then be levels above this like a building server, or the granularity could be increased below this. This gateway system will allow the system to operate over any standard network such as EIBus, Bluetooth, Lonworks and could readily be developed to allow a 'Plug N Play' allowing devices to be automatically discovered and configured using intelligent mechanisms (surprisingly, Lonworks does not have such a facility) (Homes *et al.* 2002).
In addition, it is clear such a gateway is an ideal point to implement security and data mining associated with the sub network. Figure 8 shows a logical network infrastructure in the iDorm.

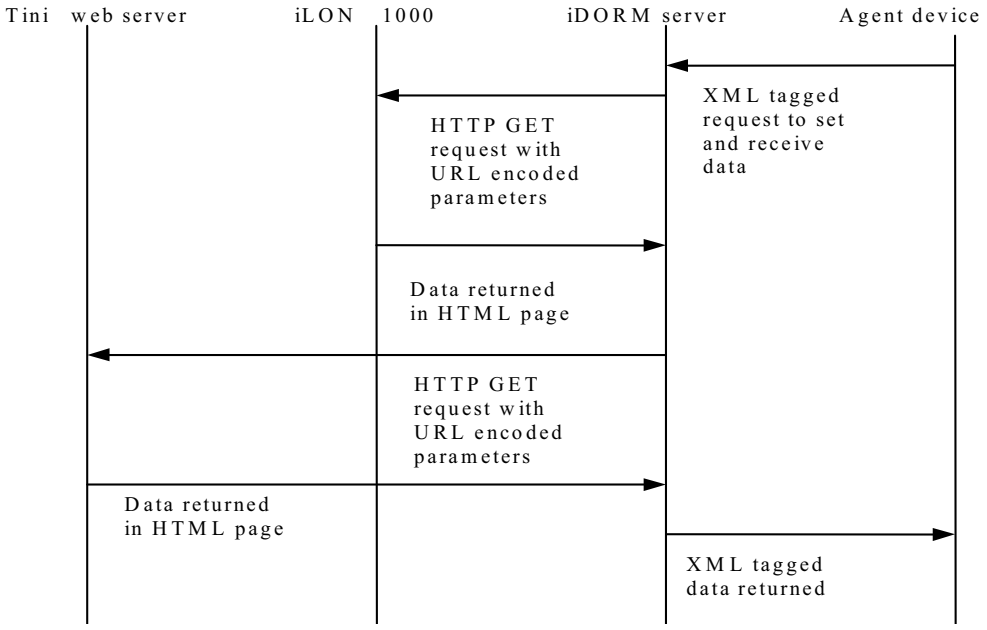


Figure 7. Diagram shows the XML based communication between the devices.

2.5.3 iDorm Interfaces

Our group has designed several interfaces to deal with the problem of being able to control the room with as few constraints as possible:

2.5.3.1 The Standard Interface

As mentioned previously, there are normal switches mounted on the walls in the iDorm that control all the effectors (lights, blind, heaters). However, these switches are not directly connected to the device they control. Each switch and button is a device on the Lonworks network. As such, it transmits a data packet across the network when it has been pressed.

2.5.3.2 The Web Interface

A small web page has been created which is accessible from any machine running a web browser. It shows the current status of the iDorm that automatically refreshes. The user can select the changes they wish to make to the environment; click on the "Update" button and the room will change. Because the web page is very simple and very small, it is possible to view it on smaller web enabled devices such as a palmtop.

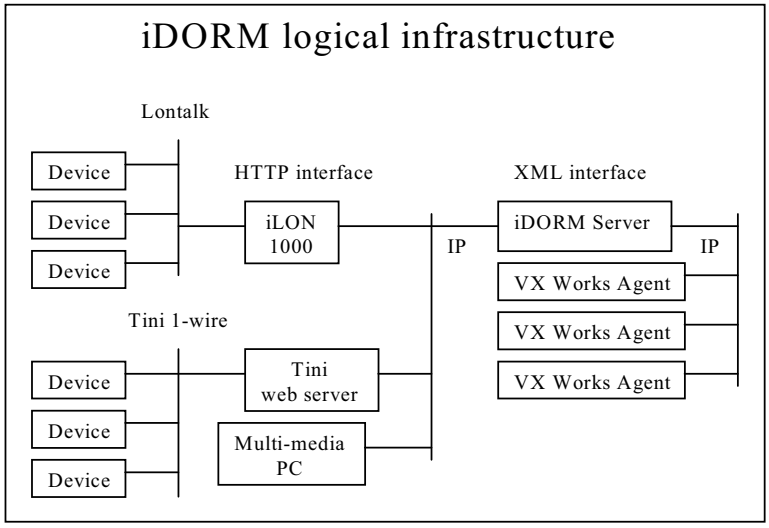


Figure 8. Diagram to show the logical network infrastructure in the iDorm.

2.5.3.3 The VRML Interface

This is a hybrid system that marries the Virtual Reality Modelling Language with a Java interface controlling the iDorm. Written in VRML 2.0, the model (depicted in Figure 9) provides a navigable virtual model of the environment that enables the user to see any part of the room and its current state. Any web browser with a VRML plug-in can view the model.

Using Java EAI, the model is linked to an applet running in the same browser window. It enables the user to choose a state for a variety of devices through a click and drag interface. Together, the model and the applet provide an information rich graphic user interface (GUI) where the user can see the current state of the iDorm both from the applet and the VRML itself (Pounds-Cornish and Holmes 2002).

The VRML runs two parallel processes, the first monitors the current state of the real world and reflects it in the model and in the interface. The second monitors the applet and notifies the agent when the user has requested a change.

The monitoring process requests a URL from the central server, which executes a TCL script to return an XML parsed document detailing the current state of the iDorm. It then parses this document and places all the pertinent values into a local vector. The process then walks down the vector changing the VRML model to reflect its recent scan of the real world. This process is looped so the model always reflects the latest state of the real world.

The notification process waits for the user to interact with the GUI, and then picks out what object was requested to change. The process then makes a GET request to the central server, which writes to a file containing the requested change(s).

When the agent is monitoring the human, it monitors the state of this file and when it spots

the user's request it takes a snapshot of the current environment state and associates the request with it. The agent then allows the request through to the appropriate device. Because of the speed of communication, the delay in the user's request is negligible but the agent is still able to learn the circumstances that cause the user to change the environment.
The advantage of the iDorm GUI 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 (Pounds-Cornish and Holmes 2002). The iDorm GUI is also designed such that more than one instance of it can be run at the same time meaning that several people can monitor the environment whilst changes are being made.
One of the future development plans for the VRML model is to reduce the complexity of the GUI to enable it to run on a wireless palmtop device.

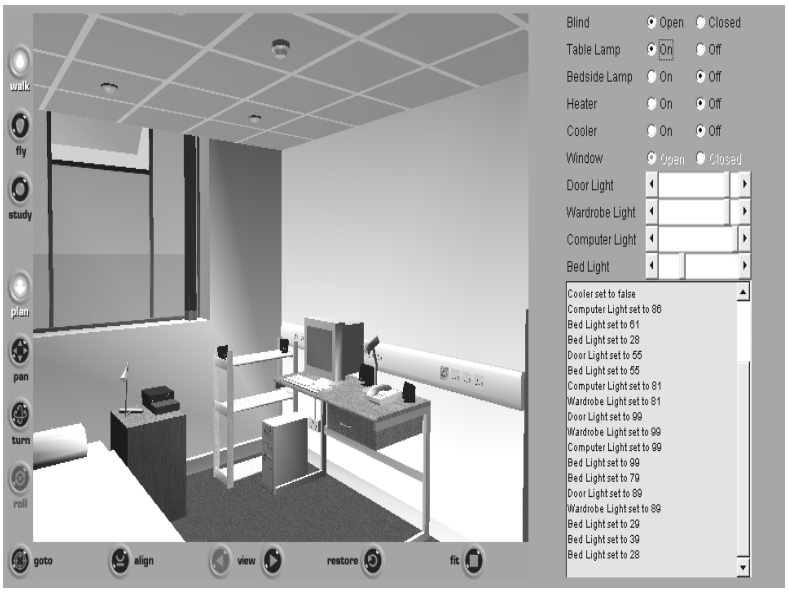


Figure 9. iDorm Graphical User Interface

2.5.3.4 WAP Interface

This interface is a simple extension of the web interface. Because the iDorm central server can also support the WML language it is possible to interact with the iDorm on mobile phones as depicted in Figure 10.

2.5.3.5 Voice Recognition Interface

Prof. Nikola Kasabov and Waleed Abdulla (Kasabov *et al*. 1999) from the University of Otago in New Zealand originated a speaker independent voice recognition system. Our research group is applying it using a room-based command set appropriate to the iDorm.
Based on Hidden Markov Models, the system contains commands created by the user. Several of the command's behaviours are dynamic, depending on the current state of the room. For instance, the command "brighter" takes an average of the ceiling light levels, adds 10% to the value and sets the spotlights accordingly. This command means the ambient light level of the room can be controlled without having to give individual commands to each spotlight.

Figure 10. WAP Interface View.

2.6 Fuzzy Incremental Synchronous Learning (ISL) Technique

According to Kasabov (Kasabov 1998) an Intelligent Agent System (IAS) should be able to learn quickly from large amounts of data. He also sates that an intelligent system should also adapt in a real time and in an on-line mode as new data is encountered. Also the system should be able to accommodate in an incremental way any new problem solving rules, as they become known. It should be memory-based, plus possess data and exemplar storage and retrieval capacities. In addition, he says that an IAS should be able to learn and improve through active interaction with the user and the environment. It should have parameters to represent short and long term memory, age, forgetting, etc. Finally he states it should be able to analyse itself in terms of behaviour, error and success. To our knowledge, no system in the field of embedded agents operating in IIE had satisfied these criteria (Kasabov 1998).

Broadly speaking this work situates itself in the recent line of research that concentrates on the realisation of artificial agents strongly coupled with the physical world. A first fundamental requirement is that agents must be grounded in that they must be able to carry on their activities in the real world, in real time (Dorigo and Colombetti 1995). Another important point is that adaptive behaviour cannot be considered as a product of an agent considered in isolation from the world, but can only emerge from strong coupling of the agent and its environment (Dorigo and Colombetti 1995). Despite this, many embedded agents researchers regularly use simulations to test their models. However, the validity of such computer simulations to build autonomous embedded agents is often criticised and is the subject of much debate. Even so computer simulations may still be very helpful in the training and testing of agents models. However as Brooks (Brooks 1992) pointed out “it is very hard to simulate the actual dynamics of the real world”. This may imply that effort will go into solving problems that simply do not come up in real world with a physical agent and that programs which work well on simulated agents will completely fail on real agents.

In this work we will refer to any learning carried out with user intervention and in isolation from the environment using simulation as *offline* learning. In our case learning will be done through interaction with the actual environment in a short time interval and we will call this

online learning. Learning the agent controllers *online* enables the learnt controller to adjust to the real noise and imprecision associated with the sensors and actuators. By doing this we can develop rules that takes such defects into account, producing a realistic controller for embedded agents, grounded in the physical world that emerge from strong coupling of the agent and its environment not in simulation. These embedded agents are grounded in the real world (situated, embodied and operating in real time), as adaptive behaviours cannot be considered as a product of an agent in isolation from the world, but can only emerge from strong coupling of the agent and its environment.

2.6.1 The Hierarchical Fuzzy Control Architecture

The methodology of Fuzzy Logic Control (FLC) appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, imprecisely or uncertainly (Pedrycz and Gomide 1998), which is the case of embedded agents operating in IIE.
As most commercial Fuzzy Logic Control (FLC) implementations feature a single layer of inferencing between two or three inputs and one or two outputs. For embedded agents, however the number of inputs and outputs are usually large and the desired control behaviours are more complex. However, by using a *hierarchical* assembly of fuzzy controllers (HFLC), the number of rules required can be significantly reduced (Saffiotti 1997). We use a variant of the method suggested by Saffiotti (Saffiotti 1997) and Tunstel (Tunstel *et al.* 1997). In this we apply fuzzy logic to both implement the individual behaviour elements and the related arbitration (allowing both fixed and dynamic arbitration policies to be implemented) (Saffiotti 1997). To achieve this we implement each behaviour as an independent FLC aimed at a simple task. The use of this hierarchical fuzzy assembly has the following advantages.

- It uses the benefits of fuzzy logic to deal with imprecision and uncertainty.
- Using fuzzy logic for the co-ordination between the different behaviours which allows more than one behaviour to be active to differing degrees thereby avoiding the drawbacks of on-off switching schema (i.e. dealing with situations where several criteria need to be taken into account). In addition, using fuzzy co-ordination provides a smooth transition between behaviours with a consequent smooth output response.
- It offers a flexible structure where new behaviours can be added or modified easily. The system is capable of performing different tasks using identical behaviours by changing only the co-ordination parameters to satisfy a different high level objective without the need for re-planning.

In general we divide the behaviours available to the embedded agent operating in IIE and specifically in the iDorm into fixed or dynamic sets, where the dynamic behaviours are learnt to achieve the person's comfort and the fixed behaviours are pre-programmed. These latter behaviours need to be predefined because they cannot easily be learnt such as the temperature at which pipes freeze or what to do in the case of fire and so on. The fixed behaviours include a safety behaviour, an emergency behaviour and an economy behaviour. The *Safety behaviour* ensures that the environmental conditions in the room are always at a safe level. The *Emergency behaviour,* which in the case of a fire alarm or another emergency, might for instance open the emergency doors and switch off the main heating and illumination systems. In the case of an emergency this will be the only active behaviour. The *Economy behaviour* ensures that energy is not wasted so that if a room is unoccupied the heating and illumination will be switched to a sensible minimum value. All of the previous behaviours are fixed but settable. For dynamic behaviours we are going to use a monitoring system, which we call an ISL to record the user actions and learn to generate rules from this information to learn his *Comfort behaviour*. These will then be

fine-tuned in an incremental and life long mode. The hierarchical fuzzy assembly is shown in Figure 11.

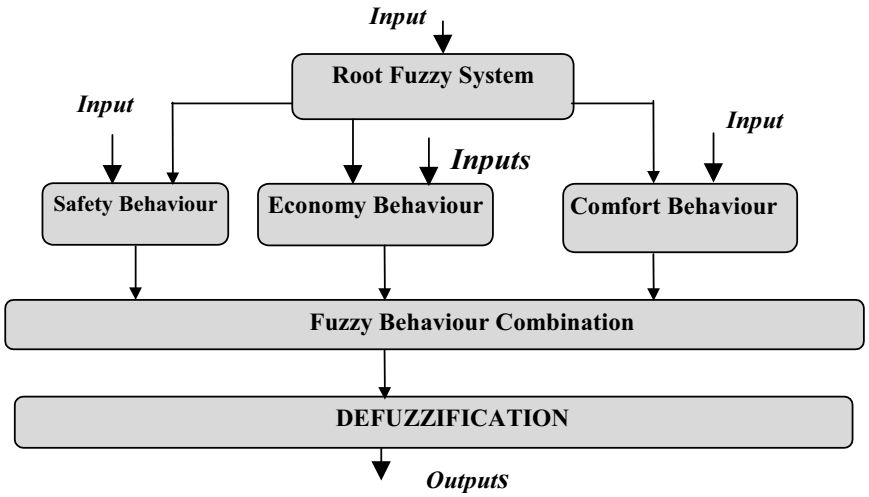


Figure 11. The Hierarchical Fuzzy Control System.

In our design each behaviour uses a FLC using singleton fuzzifier, triangular membership functions, product inference, max-product composition and height defuzzification. The selected techniques were chosen due to their computational simplicity and real-time considerations.

$$Y_t = \frac{\sum_{p=1}^{M} y_p \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}} \quad (1)$$

Where M is the total number of rules, y_p is the crisp output for each rule, $\Pi\alpha_{Aip}$ is the product of the membership functions for each rule's inputs and G is the number of inputs.
In case of using fuzzy numbers for preferences, product-sum combination and height defuzzification, the final output equation, provided by Saffiotti (Saffiotti 1997), is given below:

$$Y_{ht} = \frac{\sum_i (mm_y * y_i)}{\sum_i mm_y} \quad (2)$$

Where i represents the behaviours activated by context rules, which can be comfort, safety, emergency and economy. Y_t is the behaviour command output. mm_y is the behaviour weight which is calculated according to context rules which are suggested by the high level system and determine which behaviour is fired, and to what degree, for more information about the context rules please see (Hagras *et al.* 2000b)
The room has Eleven environmental parameters to be measured as follows:

- Time of the day (I1) measured by a clock connected to the 1-wire network represented by 4 triangular fuzzy sets (Night, Morning, Afternoon, and Evening). Note that we did not represent the time as binary sets as it very difficult to say for example that 4 am

belongs to the Night binary set and not to Morning binary sets, we think that 4 am belongs to both fuzzy sets but to different degrees. Also as the seasons change the differentiation between the sets changes, this is why we think it is more natural to set the time sets to be fuzzy sets to deal smoothly with season and timing changes and not having the abrupt changes that appears in the binary sets. Also note that the Afternoon fuzzy sets doesn't represent the time after 12 midday only but represents the midday period. The input Membership Function for the time input is shown in Figure 12

- Inside room light level (I2) measured by indoor light sensor connected to the Lonworks network represented by 3 triangular fuzzy sets (Dark Dim Bright)
- Outside outdoor light level (I3) measured by an external weather station connected to the 1-wire network represented by 3 triangular fuzzy sets (Dark Dim Bright). The input MF for both the inside and outside light levels is shown in Figure 13.
- Inside room temperature (I4) measured by redundant sensors connected to the Lonworks and the 1-Wire networks represented by 3 triangular fuzzy sets (Cold Temperate Warm)
- Outside outdoor room temperature (I5) measured by external weather station connected the 1- wire network represented by 3 triangular fuzzy sets (Cold Temperate Warm). The input MF for the inside and outside temperature is shown in Figure 14.
- Wither the user is using his audio entertainment (I6) on his computer by either he is listening to the radio or the CD player sensed by Visual C++ code when running the Winamp program represented by two binary states (Listening, Not Listening)
- Whether the user is lying or sitting on the bed or not (I7) measured by a pressure pad connected to the 1-wire network represented two binary states (On bed, Not on bed)
- Whether the user is sitting on the desk chair or not (I8) measured by a pressure pad connected to the 1- wire network represented by two binary states (On desk, not on desk)
- Whether the window is opened or closed (I9) measured by a reed switch connected to the Tini-1 Wire network represented two binary states (Open, Close)
- Whether the user is working or not (I10) sensed by a Visual C++ code that senses if the user is working on a Winword document represented two binary states (Working, Not Working)
- Whether the user is using his video entertainment (I11) on his computer by either he is watching a TV program via Wintv program or he is watching a DVD using the Winamp program, this is sensed using Visual C++ code represented two binary states (Watching, Not watching)

There are ten outputs to control;

- Fan Heater (O1) represented by ON-OFF Binary states
- Fan Cooler (O2) represented by ON-OFF Binary states.
- A dimmable spotlight above the Door (O3) represented by five triangular fuzzy sets (VLow, Low, Medium , High, VHigh).
- A dimmable spot light above the Wardrobe (O4) represented by five triangular fuzzy sets (VLow, Low, Medium, High, VHigh).
- A dimmable spot light above the Computer (O5) represented by five triangular fuzzy sets (VLow, Low, Medium, High, VHigh).
- A dimmable spot light above the Bed (O6) represented by five triangular fuzzy sets (VLow, Low, Medium, High, VHigh).
- a Desk Lamp (O7) represented by ON-OFF Binary states
- a Bedside Lamp (O8) represented by ON-OFF Binary states;
- Wither the automatic blinds are opened or closed (O9) represented by two Binary states (Open, Closed)

- If the automatic blinds are closed their opening can be controlled (O10) represented by 5 triangular fuzzy sets, where the fuzzy sets VLow, Low, deal with blind opening to the left and VHigh, High deal with blind opening to the right and Medium is 50 % opening. The output MF for the dimmable lights and the blind opening is shown in Figure 15.

This forms a rule base of 4*3*3*3*3*2*2*2*2*2*2*2 = 20736 possible rules. Which is a massive number of rules requiring large storage space and delaying the fuzzy system as each operation the system is required to perform the calculation over all these rules. We will show how the ISL will optimise this rule-base, reducing them to only those the user needs whilst allowing the ISL to add, delete and modify rules.

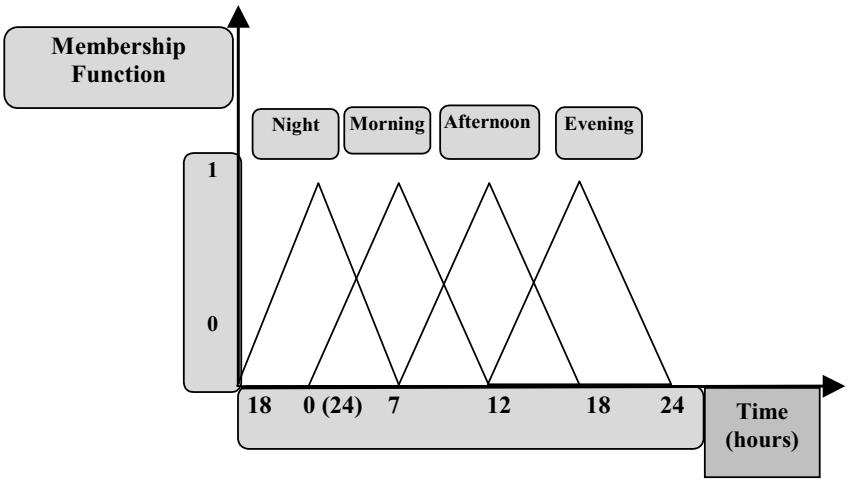


Figure 12. The input membership function of the time input

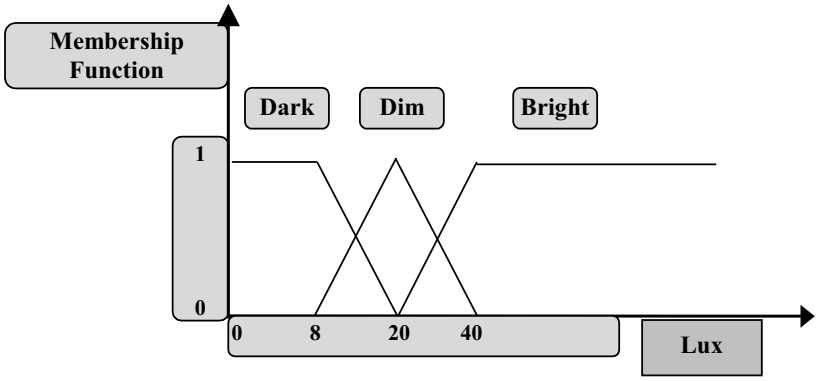


Figure 13. The input membership function of the inside and the outside light levels.

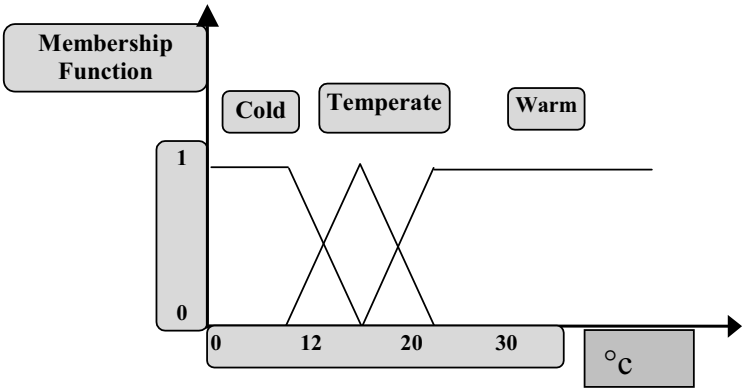


Figure 14. The input membership function of the inside and the outside temperatures.

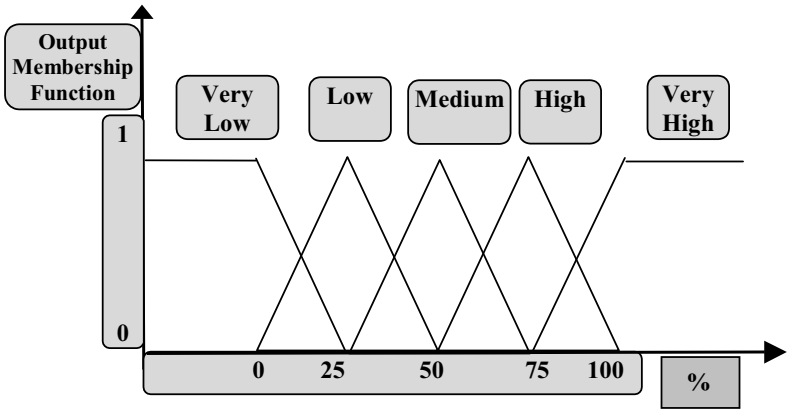


Figure 15. The output membership function of the dimmable spot lights and the blind opening.

2.6.2 The Incremental Synchronous Learning Description

The Incremental Synchronous Learning (ISL) architecture is shown in Figure 16. The embedded-agent needs to cater for somewhat different learning needs, on the one hand, short "initialisation" and on the other long "life-long" learning. In general a learning mechanism within an IIE would be life-long and non-intrusive. The ISL forms the learning engine within the control architecture and is the subject of British patent 99-10539.7. The agent is an augmented behaviour based architecture, which uses a set of parallel Fuzzy Logic Controllers (FLC), each forming a behaviour. The behaviours can be fixed or dynamic as explained above.

Each dynamic FLC (the comfort behaviour in the iDorm case) has one parameter that can be modified which is the *Rule Base* (RB) for each behaviour. Also, at the high level the co-ordination parameters can be learnt (Hagras *et al.* 2000a). The ISL system aims to provide life-long learning and adapts by adding modifying or deleting rules. It is also memory based in that it has a memory enabling the system to use its previous experiences (held as rules) to narrow down the search space and speed up learning.

The ISL works as follows:- when a new user enters the room he is identified by the active key button shown in Figure 3 and the ISL enters a Monitoring initialisation mode where it learns the users preferences during a non intrusive cycle. In the Experimental set-up we used a period of 30 minutes but in reality this is linked to how quickly and how complete we want the initial rule base. For example in a care house we want this rule base to be as complete as possible with some fine tuning, in a hotel we want this initialisation period to be small to allow fast learning. The rules and preferences learnt during the Monitoring mode form the basis of the user rules which are retrieved whenever the user renters the room. During this time the system monitors the inputs and users action and tries to infer rules from the user monitored actions. The user will usually act when given an input vector the output vector is unsatisfactory to him. Learning is based on negative reinforcement, as the user will usually request a change to the environment when he is dissatisfied with it.

After the Monitoring initialisation period the ISL enters a Control mode in which it uses the rules learnt during the Monitoring mode to guide its control of the rooms effectors. Whenever a user behaviour changes, so he needs to modify, add or delete any of the rules in the rule base the ISL goes back to the non intrusive cycle and tries to infer the rule base change to determine the users preferences in relations to the specific components of the rule that has failed. This is a very short cycle that the user is essentially unaware of and is distributed through the life-time of the use of environment, thus forming a life-long learning phase.

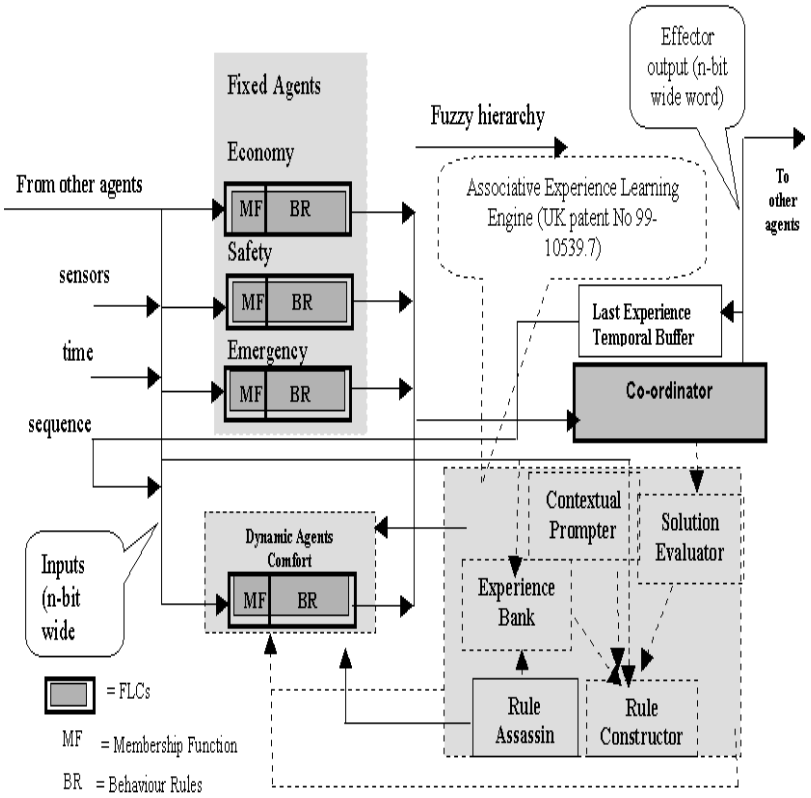


Figure 16. The ISL Embedded-Agent Architecture

As in the case of classifier systems, in order to preserve the system performance the learning mechanism is allowed to replace a subset of the classifiers (the rules in this case). The worst m classifiers are replaced by the m new classifiers (Bonarini 1999). In our case we will change all the consequences of the rules whose consequences were unsatisfactory to the user. We find these rules by finding all the rules firing at this situation where $\Pi\alpha Ai > 0$. We replace these rules consequents by the fuzzy set that has the highest membership of the output membership function. We have done this replacement to achieve the non-intrusive learning and to avoid direct interaction with the user. The learnt consequent fuzzy rule set is guided by the *Contextual prompter* which uses the sensory input to guide the learning.

The crisp output Y_t can be written as in (1). If the agent has N output variables, then we have Y_{tN}. The normalised contribution of each rule p output (Y_{pN}) to the total output Y_{tN} can be denoted by S_{rN} and is given by:

$$\mathbf{S_{rN}} = \frac{Y_{pN} \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}} \Big/ Y_{t1} \tag{3}$$

During the non-intrusive monitoring and life-long learning phases the agent is introduced to different situations, such as having different temperature and lighting levels inside and outside the room with the agent guided by the occupants desires as it attempts to discover

the rules needed in each situation. The learning system consists of learning different episodes; in each situation only small number of rules will be fired. The model to be learnt is small and so is the search space. The accent on local models implies the possibility of learning by focusing at each step on a small part of the search space only, thus reducing interaction among partial solutions. The interaction among local models, due to the intersection of neighbouring fuzzy sets means local learning reflects on global performance (Bonarini 1999). So we can have global results coming from the combination of local models, and smooth transition between close models. By doing this we don't need to learn the whole 20736 rules at once but we learn only the rules needed by the user during the different episodes. It is necessary to point to a significant difference in our method of classifying or managing rules, rather than seeking to extract generalised rules we are trying to define particularised rules.

After the initial initialisation monitoring phase the system then tries to match the user derived rules to similar rules stored in the *Experience Bank* that were learnt from other occupiers. The system will choose the rule base that is most similar to the user-monitored actions. The system by doing this is trying to predict the rules that were not fired in the initialisation session thus minimising the learning time as the search is starting from the closest rule base rather than starting from random. Also this action will be satisfactory for the user as the system starts from a similar rule-base then fine-tuning the rules.

After this the agent will be operating with the rules learnt during the monitoring session plus rules that are dealing with uncovered situations during the monitoring process which are ported from the rule base of the most similar user, all these rules are constructed by the *Rule Constructor*. The system then operates with this rule-base until the occupant's behaviour indicates that his needs have altered which is flagged by the *Solution Evaluator* (i.e. the agent is event-driven). The system can then add, modify or delete rules to satisfy the occupant by re-entering briefly to the Monitoring mode. In this case again the system finds the firing rules and changes their consequence to the desired actions by the users. We also employ a mechanism - *learning inertia* - that only admits rules to the rule base when their use has exceeded some minimal frequency (we have used 3). One of our axioms is that "the user is king" by which we mean that an agent always executes the users instruction. In the case where commands are inconsistent with learned experience learning inertia acts as a filter that only allows the rule-base to be altered when the new command is demonstrated by its frequent use to be a consistent intention. It is in this way that the system implements a life long learning strategy. It is worth noting that the system can be monitoring for lengthy periods to learn the rules necessary for a care house. Also the system can start an accelerated (intrusive) monitoring period to learn the user behaviour fast (e.g. in a hotel room) and then switch to life long learning (non-intrusive) mode.

It is worth noting that as we are dealing with embedded agents with limited computational and memory capabilities it is very difficult to deal with a large number of possible rules in the rule base (e.g. in case of the iDorm, 20736) as this will lead to large memory and processor requirements which are not realistic in embedded agents. Therefore we set a limit on the number of stored rules to 450 (in our case, the maximum number the agent can store on the onboard memory without exceeding the memory limit or degrading the real-time performance). Each rule will have a measure of importance according to how frequently this rule is used. In calculating this *degree of importance* we also include a measure of most-recent-use. The overall *degree of importance* is the product of frequency-of-use- and period-since-last use. When the memory limit is reached the *Rule Assassin* retains rules according to the priority highest-frequency, followed by most-recently-used. If two rules share the same degree of relative rule frequency recall tie breaking is resolved by a least-recently-used rule. Although not included in the current implementation, we plan not to loose the rules that are chosen for replacement but rather store them in an external hard disk

representing the *Experience Bank* so they can be recalled when needed. This action causes the onboard memory to only store the most efficient and the frequently used rules and not delaying or degrading the real time performance of the embedded agent.
Multi-Agent coordination is supported by making compressed information available to the wider network. The compressed data takes the form of a status word describing which behaviours are active (and to what degree). As with any data, the processing agent decides for itself which information is relevant to any particular decision. Thus, multi-agent processing is implicit to this paradigm. In addition, a collection of agents can be regarded as a higher-level agent, and in turn equivalent to a single sophisticated sensor. It can readily be seen that there is both a stigmatic and recursive view involved in this mechanism giving this system a simple, elegant but scaleable independent mechanism of coordination. The leads to its name RSC (Recursive Stigmatic Coordination) Paradigm (Callaghan *et al.* 2001). We have found that receiving high level processed information from remote agents, such as "the room is occupied" is more useful than being given the low level sensor information from the remote agent that gave rise to the high-level characterisation. This is because the compressed form both relieves agent-processing overheads and reduces network loading (Colley 2001).

2.7 Performance of Embedded Agents

From what was said earlier, embedded agents have comparably little computing resources. For example, the embedded agent we use in Essex shown in Figure 17 is based on 68000 Motorola processor with 4 Mbyte of RAM and an Ethernet network connection. It runs the VxWorks Real Time Operating System (RTOS). Clearly such agents have performance limitations in that they have limited I/O (24 lines in our case) and can only support a certain number of concurrent processes within real-time computational limits (16 medium size processes, each of less than 100 lines of code, with process pre-emption times of less a maximum of 2ms, in our case). That being said typical behaviours need only a handful of lines, 20 lines of core control code being large). As our processor is relatively modest in complexity, and quite capable of being fabricated using readily available microelectronics. The specification quoted above can readily be seen to be more than adequate for most ubiquitous devices such as security systems or, lighting controllers and so forth thus it is possible to state that useable agents can be realised from available technology. In addition, as larger systems (GadgetWorlds etc) are comprised of multiple eGadget systems (most with integrated agents) then the system scales without incurring computational constraints. As explained in the processing section, the RSC inter-agent coordination system being utilised simply treats other agents as sophisticated sensors, with coordination being achieved by observation of other agent's status rather than any sophisticated interacting involving computationally intensive messaging. The stigmatic status messages transmitted by agents are infrequent (e.g. 1 per minute) and small (e.g. a few bytes). As a large GadgetWorld would be no more than 100 eGadgets it can readily be seen that inter-agent coordination scale upwards to this level with negligible overheads leaving the basic limitation as being the relationship between the basic single agent and the control system it is embodied in.

2.8 Experimental Results

We have conducted a number of experiments with various different users staying in the iDorm for different lengths of time. We have performed some experiments with a limited

number of inputs (the inside light level (I1) and outside light level (I2) and the inside temperature (I3) and the outside temperature (I4) and wither the user in on the bed or the desk (I5). The control outputs are as follows, O1 which is a row of dimmable spotlights above the desk and O2 is a row of dimmable spotlights above the bed, O3 is the Bedside Lamp, O4 is the desk Lamp, O5 is the blind opening and O6 is the fan heater and O7 is the Fan cooler

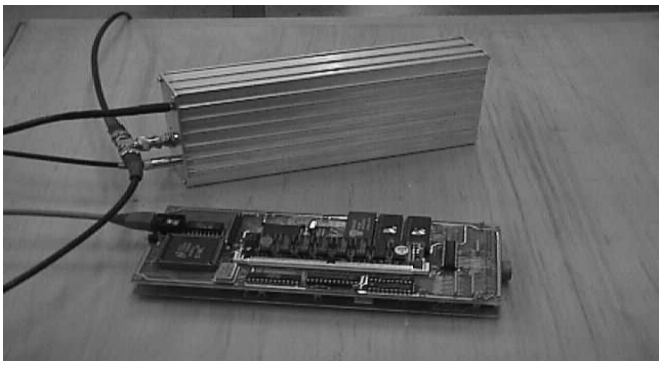


Figure 17. The Essex embedded agent.

Table 2 shows the learnt rule-base for "User-1" pictured in Figure 18 who occupied the room for more than two hours. In these experiments the user undertook many behaviours such as studying during the day (in which the lighting was bright) and studying in the evening (i.e. when external light was fading). This specific user preferred to use all the ceiling lights ON and the blind open to Norm. Another behaviour was lying in bed reading with the blind adjusted to his convenience and the bedside light sometimes being used. He would also close the blinds in the evening using combinations of the ceiling lamps and desk lamp. The experimental data also contains "going to sleep behaviours" including reading before sleeping and getting up spontaneously at night to work (they are students!).

Table 2. The learnt Rule Base for "User-1"

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7
XX	Med	Bright	Desk	Vhigh	Vhigh	OFF	OFF	Med	OFF	OFF
XX	High	Bright	Desk	Vhigh	Vhigh	OFF	OFF	Med	OFF	OFF
High	Low	Dim	Desk	Vhigh	Vhigh	OFF	OFF	Med	OFF	OFF
Med	Med	Bright	Bed	Vhigh	Vlow	ON	OFF	Vlow	OFF	OFF
Med	High	Bright	Bed	Vhigh	Vlow	ON	OFF	Vlow	OFF	OFF
High	Med	Bright	Bed	Vhigh	Vlow	ON	OFF	Vlow	OFF	OFF
High	High	Bright	Bed	Vhigh	Vlow	ON	OFF	Vlow	OFF	OFF
Med	Med	Dark	Desk	Med	Med	ON	ON	Med	OFF	OFF
Med	High	Dark	Desk	Med	Med	ON	ON	Med	OFF	OFF
High	Med	Dark	Desk	Vhigh	Vhigh	ON	OFF	Vhigh	ON	OFF
High	High	Dark	Desk	Vhigh	Vhigh	ON	OFF	Vhigh	ON	OFF
Low	Med	Dark	Bed	Vlow	Vlow	OFF	OFF	VLow	OFF	OFF
Low	High	Dark	Bed	Vlow	Vlow	OFF	OFF	VLow	OFF	OFF
Low	High	Dark	Desk	Med	Med	ON	ON	Med	OFF	OFF
Low	High	Dark	Desk	Med	Med	ON	ON	Med	OFF	OFF

The ISL learnt 15 rules of which the first 7 rules were learnt during the Initialisation phase. The next 4 rules were ported from similar users and were satisfactory to the user. The last 4 rules resulted from fine-tuning the ported rules, these rules dealt with darkness where the first two rules dealt with the user wanting to sleep and he wanted all lights off while the similar user slept with the desk lamp ON because he doesn't like darkness. The last two rules dealt with user returning to the desk to read as he couldn't sleep he switches all lights to Medium and switch ON the desk and the bedside lamp and the blind to Medium to allow more light, the similar user had the same behaviour but he was closing the blind. It is

obvious that the room user and the similar user actions are very similar and we needed only a fine-tuning to satisfy the current user needs, this is an advantage of using the *Experience Bank* which reduces the life long learning time and satisfies the user. The XX in Table 2 indicate a No-Care situation, which resulted as the inside room lighting level was not important as when it was bright the user took always the same actions. This shows also that ISL besides optimising the rule base can help to identify and focus the input parameters required by the user and hence giving more optimisation to the system.

Figure 18. "User 1 in the iDorm".

We also conducted a set of interesting experiments with the whole set of sensors and actuators discussed in Section 2.6.1 in which another room user (user 2) had occupied the room continuously for a period of 51 hours over two nights while the room was controlled by our embedded agent implementing the ISL techniques. Since the room was originally designed as a multi-purpose space, it was possible to use it both as a dormitory and a workplace. The room was treated as a standard living space in which there were no artificial constraints such as periods of occupancy or behaviour. The Agent was connected to the iDorm server and the user was given access to the VRML GUI to adjust the environment as they saw fit.
The user was identified by his intelligent key, which activates the active lock explained in Section 2.5. Figure 19 shows the user using his intelligent key to access the room.

Figure 19. User (2) accesses the room via his intelligent key.

Throughout the experimentation period, the user adjusted the environment using the VRML GUI whenever they were not happy with the current state of the environment and made a note of the decisions they were making in a journal. Whenever this action occurred, the Agent received the request, generated a new rule or adjusted a previously learnt rule and allowed the action through (see Figure 20). The agent would react to similar environmental states by taking the learnt user action.

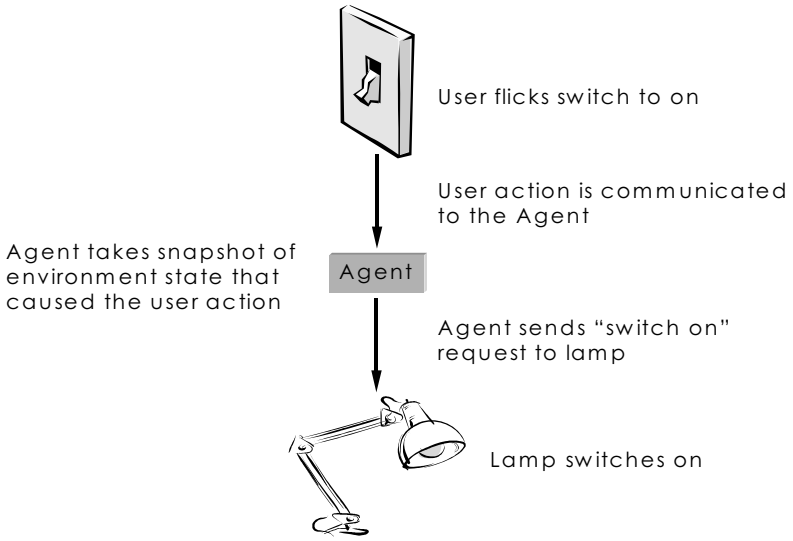


Figure 20. The Agent Communication Path

At the end of the experiment, the results returned by the agent would be a set of fuzzy rules about the user's behaviour over the previous 51 hours and a list of journal entries provided by the user.

A small parsing tool was written to convert the text file containing the fuzzy rule set into a human readable format. At the end of the experiment, the rules were converted into this form and examined in two different ways. The first involved comparing the human readable rules to the journal entries from the user to ensure that the Agent had successfully learnt the behaviours the user was exhibiting. The second was to compare the number of rules learnt over time. The Agent's success can be measured by monitoring how well it matches the environment to the user's demands. If it does well, the user will have to generate fewer rules over time. If it does badly, the user will have to generate more rules over time.

At the end of the 51-hour period, the Agent had learnt 324 rules. Table 3 gives the first ten rules learnt by the Agent and Table 4 gives the first and the eighth rules translated by the parsing software tool. In Table 3 for I1 "0" represents the Night fuzzy set while "1" represents the Morning fuzzy set and "2" represents the Afternoon fuzzy set and "3" represents the evening fuzzy set. For the inside and the outside light levels (I2, I3) "0" represents Dark fuzzy set, "1" represents the Dim fuzzy set and "2" represents the Bright fuzzy sets. For the inside and outside temperatures (I4, I5) "0" represents the Cold fuzzy set and "1" represents the temperate fuzzy sets and "2" represent the Warm fuzzy set. For O3, O4, O5, O6, O10, "0" represents the Very Low fuzzy set, "1" represents the Low fuzzy set, "2" represents the Medium fuzzy set, "3" represents the High fuzzy set and "4" represents the Very High fuzzy set. All the other inputs and outputs are represented by binary sets in which "0" is False and "1" is ON. For the blind "0" is open and "1" is closed.

Figure 21 shows the ISL activating rule 1 in Table 3 in which it is afternoon time and the user is sitting on his chair to read. Figure 22 shows the ISL output during night time when the user is sleeping in which he prefers the bed side lamp to be ON and the blinds to be closed.

Table 3. First 10 Rules Learnt by Agent

I1	I2	I3	I4	I5	I6	I7	I8	I9	I 10	I 11	O 1	O 2	O 3	O 4	O 5	O 6	O 7	O 8	O 9	O 10
2	2	2	1	1	0	0	1	1	0	0	0	0	0	0	4	0	0	0	0	0
2	2	2	1	2	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
2	2	2	2	1	0	0	1	1	0	0	0	0	0	0	4	0	0	0	0	0
2	2	2	2	2	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
3	2	2	1	1	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
3	2	2	1	2	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
3	2	2	2	1	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
3	2	2	2	2	0	0	1	1	0	0	0	1	0	0	4	0	0	0	0	0
2	2	2	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	2	2	1	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Table 4. The Parsed First and Eighth Rules Learnt by Agent

In the afternoon when it is bright inside and bright outside, temperate inside and temperate outside, sitting on the chair, with the window open I switch the heater off, switch the cooler off, switch the door spotlight Very Low, switch the wardrobe spotlight Very Low, switch the computer spotlight Very High, switch the bed spotlight Very High, turn off the bed lamp turn off the table lamp and open the blind.
In the evening when it is bright inside and bright outside, warm inside and warm outside, sitting on the chair, with the window open I switch the heater off, switch the cooler on, switch the door spotlight Very low, switch the wardrobe spotlight Very Low, switch the computer spotlight Very High, switch the bed spotlight Very Low, turn off the bed lamp turn off the table lamp and open the blind.

Figure 21. User (2) is sitting at the desk in the afternoon in the iDorm, which is controlled by the ISL.

Figure 22. User (2) is sleeping in the bed in the iDorm, which is controlled by the ISL.

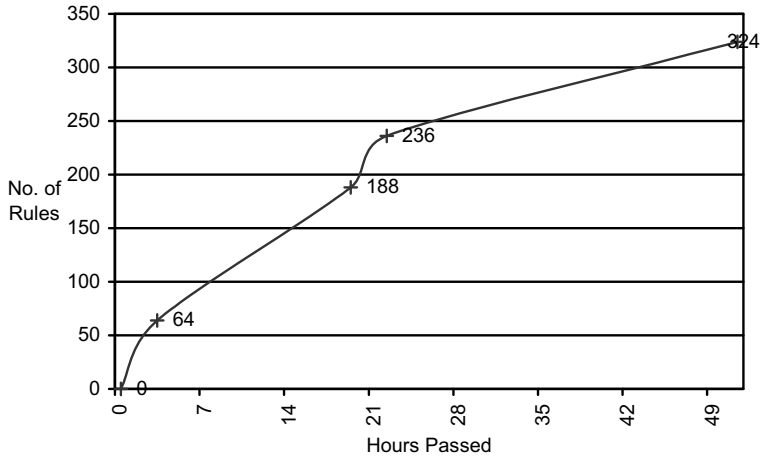


Figure 23. Rules Learnt by Agent Over Time

The agent had learnt the 324 rules needed to capture the behaviour of this user over the 51-hour experiment, which demonstrated that our system can learn effectively using the ISL and it doesn't need to learn the whole 20736 to act optimally. Figure 23 shows the number of rules learnt over the duration of the experiment. Figure 23 suggests that the Agent had to learn less new rules about the user as the experiment progressed; the latter was one of our criteria for measuring the Agent's success. Using the evidence of the continual reduction in the learning rate, we can conclude that the Agent managed to pick out the pertinent behaviour of the user over time. We can also conclude from matching the journal entries to the learning rate that the agent adapts the size of its rule set to the frequency of user behaviour. That is to say, 21 hours into the experiment there is a significant rise in the learning rate. This can be matched to mid-morning where the user leaves the iDorm for a coffee break and changes the state of a number of devices before leaving the room. If the agent didn't remember this behaviour then at the same time the following day (45 hours into the experiment), there would be a similar sharp rise where the user repeated the same behaviour. However, it can be seen from the graph that the Agent's learning rate is unaffected at this time suggesting that the user was content with the Agent's behaviour.

These experiments had offered surprising results in terms of the little information the agent had to gather in order to autonomously create a comfortable environment with diminishing

need for user correction. Based on these results, it is feasible for the research team to give the Agent access to more of the input vector – increasing the complexity of the behaviour it can monitor. These learning techniques are very important for embedded agents operation in IIE populated by a large number of eGadgets.

2.9 Conclusions

In this chapter we have presented our Incremental Synchronous Learning (ISL) mechanism for online learning and adaptation of embedded-agents embodied in intelligent inhabited environments populated with eGadgets (which we have developed as an exemplary ubiquitous computing model). This work is part of the EU Disappearing Computer programmes eGadgets project. The main goal of this project is to support user-driven design of ad-hoc assemblies of a computer based artefacts that will make up many envisaged ubiquitous computing environments. Embedding useful amounts of intelligence into eGadgets environments was seen as an essential enabling technology to achieve the vision of the eGadgets project. In particular, it solves the problem of how would an eGadget (or other ubiquitous computing device) adapt its control to whatever ad-hoc set of connections a user decided to provide a particular device with.

We have also discussed how transferring some cognitive capabilities from people into artefacts is a natural way to facilitate the disappearance of computers as computers are increasingly embedded into our daily environment. We have also argued that embedded-intelligence can bring significant cost and effort savings over the evolving lifetime of product by avoiding expensive programming (and re-programming). In particular, if people are to use collections of computer based artefacts to build systems to suit their own personal tastes (which may be unique in some sense) then self programming embedded-agents offer one way of allowing this without incurring an undue skill or time overhead.

Our techniques were evaluated in the Essex iDorm which is an intelligent dormitory that makes an excellent evaluation platform for ubiquitous computing and ambient intelligence work as it provides a compact multi-use space with occupant that are sympathetic to exploring new technology. We had carried unique experiments in which the iDorm has been occupied by various people and up to two day of continuous occupancy. Our fuzzy logic based ISL had demonstrated the capability of the method to provide online learning in both set-up and life long learning cycles. We have demonstrated a novel feature of this agent in that it particularises itself to the users behaviour (including idiosyncratic actions) rather than to the machine or by generalising for a group of users.

For our current and future work we have plans to conduct more and longer experiments with the iDorm (up to a year to get a full climatic cycle), significantly expand the sensor-effector set and explore more fine-grained and course grained distributed embedded-agents (e.g. with agents in eGadgets, communities of eGadgets forming GadgetWorlds and even inter-communicating rooms). We are also investigating the integration of mobile agents such as robots (Colley *et al.* 2001) and wearable agents (e.g. cellphones, watches etc).

Acknowledgements

We are pleased to acknowledge the funding support from the EU IST Disappearing Computer program (eGadgets) and the Korean-UK Scientific Fund programme (cAgents). We are also pleased to acknowledge our eGadget partners Kieran Delaney (NMRC), Achilles Kameas, Irene Mavrommati and Manolis Koutlis (CTI) and our cAgents partners from KAIST Professor Zenn Bien and Mr. Kim and Mr. Lee and Mr. Myung whose numerous and challenging scientific discussions have contributed to our thinking in this area.

References

Angelov, P., Buswell, R., Hanby,V. (2000), " Automatic Generation of Fuzzy Rule-based Models from Data by Genetic Algorithms, *Proceedings of International Conference on Recent Advances on Soft Computing, Leicester, UK*.

Bonarini, A. (1999), "Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier systems", *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 52-60.

Brooks, R. (1992), "Artificial Life and Real Robots", MIT press.

Brooks, R. (1997), "Intelligent Room Project", *Proceedings of the 2nd International Cognitive Technology Conference (CT'97)*, Japan.

Callaghan, V., Clarke, G., Pounds-Cornish, A. (2000a) "Buildings As Intelligent Autonomous Systems: A Model for Integrating Personal and Building Agents", *The 6th International Conference on Intelligent Autonomous Systems (IAS-6), Venice, Italy.*

Callaghan, V., Clarke, G., Colley, M., Hagras, H. (2000b), " Embedding Intelligence: Research Issues for Ubiquitous Computing", *Proceedings of the Ubiquitous Computing in Domestic Environments Conference, Nottingham-UK.*

Callaghan, V., Clarke, G., Colley, M., Hagras, H. (2001), "A Soft-Computing based DAI Architecture for Intelligent Buildings" *Studies in Fuzziness and Soft Computing on Soft Computing Agents, Physica-Verlag-Springer.*

Colley, M., Clarke, G., Hagras, H., Callaghan V. (2001), "Intelligent Inhabited Environments: Co-operative Robotics & Buildings", *32nd International Symposium on Robotics (ISR 2001), Seoul, Korea.*

Davisson, P. (1998), "Energy Saving and Value Added Services; Controlling Intelligent-Buildings Using a Multi-Agent System Approach" *in DA/DSM Europe DistribuTECH, PennWell*.

Dorigo M., Colombetti, M. (1995), "Robot Shaping: Developing Autonomous agents through learning", *Artificial Intelligence Journal*, Vol (71), pp. 321-370.

Hagras, H., Callaghan, V., Colley, M. (2000a),"Learning Fuzzy Behaviour Co-ordination for Autonomous Multi-Agents Online using Genetic Algorithms & Real-Time Interaction with the Environment " *Proceedings of the 2000 IEEE International Conference on Fuzzy Systems, San Antonio-USA*, pp. 853-859.

Hagras, H., Callaghan, V., Colley, M., Clarke, G. (2000b) "A Hierarchical Fuzzy Genetic Agent Architecture for Intelligent Buildings Sensing and Control", *Proceedings of the International Conference on Recent Advances in Soft Computing, Leicester, UK*.

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

Kasabov, N. (1998), “Introduction: Hybrid intelligent adaptive systems", *International Journal of Intelligent Systems,* Vol.6, pp.453-454.

Kasabov, N., Kozma, R., Kilgour, R., Laws, M., Taylor, J., Watts, M. (1999), "A. Hybrid connectionist-based methods and systems for speech data analysis and phoneme-based speech recognition". In: *Neuro-Fuzzy Techniques for Intelligent Information Processing*, N. Kasabov and R.Kozma, Eds. Heidelberg, Physica Verlag.

Minar, N., Gray, M., Roup, O., Krikorian, R., Maes, P. (1999), “HIVE: Distributed Agents for Networking Things”. MIT Media Lab, *Appeared in ASA/MA*.

Mozer, M. (1998), “The Neural Network House: An Environment That Adapts To Its Inhabitants”, *Proceedings of American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, AAAI Press,* pp. 110-114.

Pedrycz, W., Gomide, F. (1998), “ An Introduction to Fuzzy Sets: Analysis and Design”, MIT press, Cambridge.

Pounds-Cornish, A., Holmes, A. (2002), "The iDorm - a Practical Deployment of Grid Technology" *Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid* (CCGrid2002), Berlin, Germany.

Saffiotti, A. (1997), " Fuzzy Logic in Autonomous Robotics: Behaviour Co-ordination" Proceedings of the 6th IEEE International Conference on Fuzzy Systems, Spain , pp. 573-578.

Sharples, S., Callaghan, V., Clarke, G. (1999), “A Multi-Agent Architecture For Intelligent Building Sensing and Control”, International Sensor Review Journal, Vol. 19. No. 2.

Tunstel, E., Lippincott, T., Jamshidi, M. (1997), "Behaviour Hierarchy for Autonomous Mobile Robots: Fuzzy Behaviour Modulation and Evolution", *International Journal of Intelligent Automation and soft computing, Vol .3, pp. 37-49*.