IEEE Intelligent Systems, Vol. 19, Issue 6, pp. 12-20, November/December 2004

Creating an Ambient-Intelligence Environment Using Embedded Agents

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

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK.

Abstract— We outline a vision, methodology and architectural design for creating an ambient environmental intelligence within a ubiquitous computing environment made up of scaleable, networked embedded agent-based artefacts. Embedded-agents are integrated into artefacts, such as domestic appliances or robots to give the artefact some useful amount of intelligence. We describe a practical ubiquitous computing test-bed environment, the Essex intelligent Dormitory (iDorm), which combines a number of heterogeneous artefact based agents and networks. We show novel results where the iDorm was occupied by a user for five and half consecutive days. During this period the iDorm was monitored and controlled by an intelligent and adaptive embedded agent using our fuzzy Incremental Synchronous Learning (ISL) approach. The ISL operates in a life long learning mode realising ambient intelligence in which it learns the user behaviour and controls the room according to his preferences in a personalised and non-intrusive way.

Keywords: Fuzzy Logic, Learning, Embedded Agents, Ambient Intelligence, Ubiquitous Computing, Pervasive Computing

Ambient Intelligence in Ubiquitous Computing Environments

"Star Trek" and similar science fiction films and television series paint an intriguing picture of the future, one in which masses of unseen and tireless electronic devices and intelligent-agents attend to the occupants of their space habitats every need; regulating the air they breath, the temperature of their cabins, their entertainment and communications, etc.. For many, space exploration and extraterrestrial planetary habitats are not just the "final frontier" for mankind but the ultimate vision for ubiquitous computing and ambient intelligence.

Ubiquitous computing is a paradigm where technology becomes virtually invisible. Instead of having a desktop or laptop machine, the technology we use is embedded in our environment [11]. Recent estimates show that of the order of 8 billion microprocessors that were produced in 2001, only 2% of which went into PCs, the rest went into embedded computer devices most people wouldn't recognize as computers [3]. Cheap and compact microelectronics means everyday artefacts (e.g. clothing, desks) are now potential targets of embedded-computers, while ever-pervasive networks allow such artefacts to be associated together in both familiar and novel arrangements to make highly personalized systems. This new and exciting computing paradigm promises to revolutionize the way we interact with computers, services, and the surrounding physical spaces, yielding higher productivity and a more seamless interaction between users and computing services. To fully realise this vision ambient intelligence is needed.

Ambient intelligence refers to an exciting new information technology paradigm where people are empowered through a digital environment that is aware of their presence and context, and is sensitive, adaptive, and responsive to their needs. Ambient intelligence improves the quality of life by creating the desired environment conditions and functionality via intelligent, personalised interconnected systems and services. Ambient intelligence environments are characterised by their ubiquity, transparency and intelligence. Ubiquitous because the user is surrounded by a multitude of inter-connected embedded systems. Transparent because the computing equipment appears invisible to the user as it is seamlessly integrated into the background. Intelligent because the system is able to recognise the people that live in it and is able to program itself to meet their needs by learning from their behaviour.

For the vision of ambient intelligence to be realised, people must be able to use and configure computer-based artefacts and systems without being cognitively overloaded. If the user has to program each device and work out how to connect them together to achieve the required functionality, this vision may never be realised. In many computer-based products the computer remains very evident and the user is forced to refer to complicated manuals and to use their own reasoning and learning processes to use the device successfully. This situation is likely to get much worse as the number, varieties and uses of computer based artefacts increases. The increasing complexity leads to the need to design a system that would allow intelligence to disappear into the infrastructure, automatically learning to carry out everyday tasks based upon the users' habitual behaviour. Such a system would enable devices embedded in the environment to co-operate with one another to make a wide range of new and useful applications, to achieve greater functionality, flexibility and utility. Such a system would allow the computational resources to disappear into the infrastructure to define active spaces [3]; buildings, shopping malls, theatres, domestic environments, instrumented with embedded devices that collaborate in response to

users' behaviour to automatically carry out everyday tasks. The compelling question then becomes "how can we achieve the vision of ambient intelligence in such ubiquitous computing environments?" One approach, which we present in this paper, is to embed intelligent agents into the devices that make up ubiquitous environments thus forming *embedded agents*.

The work proposed by this paper focuses on the development of learning and adaptation techniques for embedded agents seeking to provide an online, life-long, personalised learning of anticipatory adaptive control to realise ambient intelligence in ubiquitous computing environments. We will use the Essex intelligent dormitory (iDorm) as a testbed for our work

Intelligent Embedded Agents

Embedded intelligence can be regarded as the inclusion of some capacity for reasoning, planning and learning in an artefact. Embedded-computers that contain such an intelligent capability are normally referred to as "*embedded-agents*" [3].

It is now common for such "*embedded-agents*" (as intrinsic parts of "*intelligent artefacts*") 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.

Most automation systems (which involve a minimum of intelligence) utilize mechanisms that generalize actions (e.g. set temperature or loudness that is the average of many peoples' needs). However, we argue that Artificial Intelligence (AI) applied to personal artefacts and spaces needs to *particularize* to the individual [3]. Further, we argue that it is essential that any agent serving a person should always and immediately carry out any requested action (i.e. people are always in control, subject to overriding safety considerations). The embedded-agent learning technique we will outline is characterized by its ability to particularize its actions to individuals *and* immediately execute commands, wherever that is a practical possibility. Thus, the value of an intelligent embedded agent lies in the agent's ability to learn and predict the human and the system needs, automatically adjusting the agent controller based on a wide set of parameters [2]. There is thus a need to modify effectors for environmental variables such as 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 affecting 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. 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.

Intelligent Inhabited Environments and Intelligent Buildings

Intelligent Inhabited Environments (IIE) are spaces such as cars, shopping malls, homes and even our own bodies that will allow these environments to respond "thoughtfully" to our needs. Such environments would consist of a multitude of, possibly disconnected active spaces to provide ubiquitous access to system resources according to the current context of the user. Such environments promise a future where computation will be freely available everywhere, in a similar fashion to the current availability of batteries and power sockets. Computation will constitute the human world, handling our goals and needs. Devices, either handheld or embedded in the environment, will bring computation to us, no matter where we are or in what circumstances. These devices will personalize themselves in response to our presence and behaviour. Precursors to such environments can be found now in intelligent buildings

A typical 'container' environment for ubiquitous computing is an intelligent building which might be a house or office. The heterogeneity, dynamism and context-awareness in a home make it a good choice to explore design challenges within the range of ubiquitous systems. We view intelligent buildings as computer-based systems, gathering information from a variety of sensors (and other computers), and using intelligent embedded-agent techniques to determine the appropriate control actions. In controlling such systems one is faced with the imprecision of sensors, the large number of information sources, lack of adequate models of many of the processes and the non-deterministic aspects of human behaviour. It is important for the building that embedded agents are able to continuously learn and adapt to the needs of the individuals within, whilst always providing a safe and timely response to any situation.

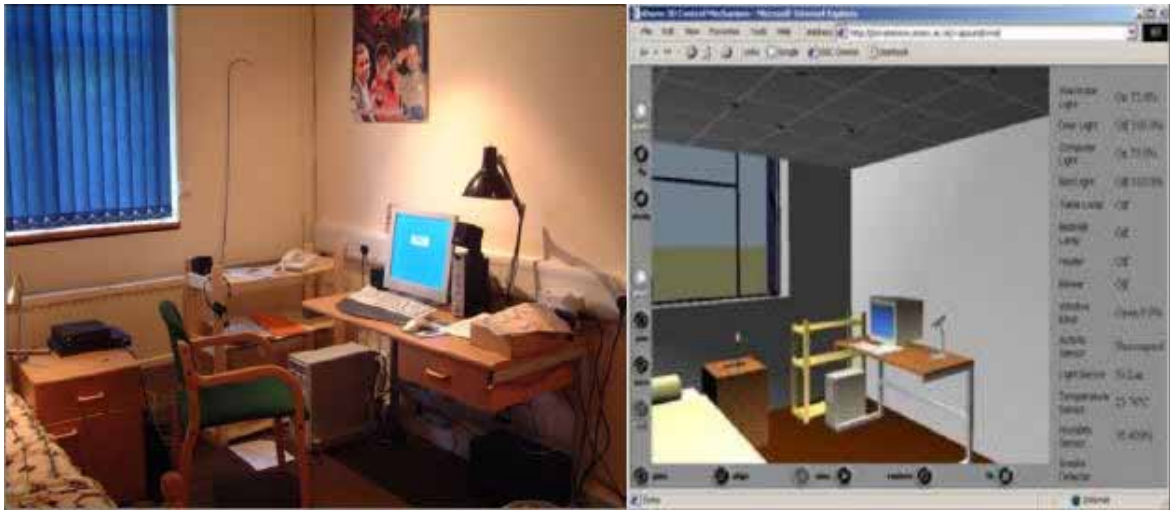
There are a growing number of research projects concerned with applying intelligent agents to IIE and intelligent buildings. In Sweden, Davidsson [3] utilises multi-agent principles to control building services. These agents are based on the AI thread that decomposes systems by function rather than behaviour as in our research. Their work does not address issues such as occupant based learning. In Colorado Mozer [9] 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 is sometimes at the expense of the occupant's comfort.

Work at MIT on the HAL project [2] concentrates on making the room responsive to the occupant by adding intelligent sensors to the user interface. Context Aware systems such as that of the Aware Home work at Georgia Tech are more concerned with time independent context rather than temporal history or learning as is a central issue in this proposal [1]. There are other high profile Intelligent Environment projects such as the Microsoft Smart House, BT's Tele-care and Cisco's Internet Home [10]. However most of these industrial projects, including home automation technologies, like Lonworks and X10 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 authors' knowledge no other work has addressed the online learning and adaptation of intelligent embedded agents to the users' habitual behaviour operating within IIE [6].

The iDorm – A Testbed for Ubiquitous Computing and Ambient Intelligence

We have chosen the Essex Intelligent Dormitory (iDorm) pictured in Figure (1-a) to be a demonstrator and test-bed for ubiquitous computing environments and use it to test our intelligent learning and adaptation mechanisms needed by the embedded agents for the realization of ambient intelligence in ubiquitous computing environments. 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 one room apartment for elderly or disabled people or an intelligent hotel room. The room, which looks like any other room, contains the normal mix of furniture found in a study/bedroom allowing the user to live comfortably. The furniture (most of which are fitted with sensors that provide data for the network) includes a bed, a work desk, a bedside cabinet, a wardrobe and a PC-based work and multimedia entertainment system. The PC contains most office type programs to support work and the entertainment support includes audio entertainment (e.g. playing music CDs and radio stations using Dolby 5.1 surround sound) as well as video services (e.g. television and DVDs, etc).

In order to make the iDorm as responsive as possible to the needs of the occupant it is fitted it with an array of embedded sensors (e.g. temperature, occupancy, humidity, light level sensors etc) [8] and effectors (e.g. door actuators, heaters etc). Amongst the many interfaces, we have produced a virtual reality system (VRML) shown in Figure (1-b) that marries the Virtual Reality Modelling Language with a Java interface controlling the iDorm. It provides the user with a visualisation tool showing the current state of the iDorm and allows direct control of the various effectors in the room.



(a) (b)

Figure (1): a) Photo of iDorm. b) The iDorm VRML interface.

Although the room looks like any other, above the ceiling and behind the walls hides a multitude of different networks and networked devices. In building the iDorm, we have installed devices that reside on several different types of network. As such access to the devices needs to be managed, gateways between the different networks *can be regarded as critical components in such systems*, combining appropriate granularity with security. Currently the iDorm is based around three networks Lonworks, 1-Wire (TINI) and IP, providing the diverse infrastructure present in ubiquitous computing environments and allowing the development of network independent solutions [8].

Lonworks is Echelon's proprietary network and encompasses a protocol for building automation. There are many commercially available sensors and actuators for this system. The physical network installed in the iDorm is the 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.

The 1-Wire network, developed by Dallas semiconductor was designed for simple devices to be connected over short distances. It offers a wide range of commercial devices including small temperature sensors, weather stations, ID buttons and switches. The 1-Wire network is connected to a Tiny Internet Interface board (TINI board), which runs an embedded web server serving 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 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 work and entertainment in the iDorm. Again the MMPC uses the HTTP protocol to display its information as a web page.

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 by using a hierarchy of gateways it would be possible to create a scaleable architecture across such heterogeneous networks in IIE [8]. The iDorm gateway server allows a standard interface to all of the room's sub networks by exchanging XML formatted queries with the entire principal computing components, which overcomes many of the practical problems of mixing networks.. This gateway system will allow the system to operate over any standard network such as EIBus, Bluetooth and Lonworks and could readily be developed to include 'Plug N Play' allowing devices to be automatically discovered and configured using intelligent mechanisms (as far as we know, Lonworks does not have such a facility) [8]. In addition, it is clear such a gateway is an ideal point to implement security and data mining associated with the sub network. Figure (2) shows a logical network infrastructure in the iDorm.

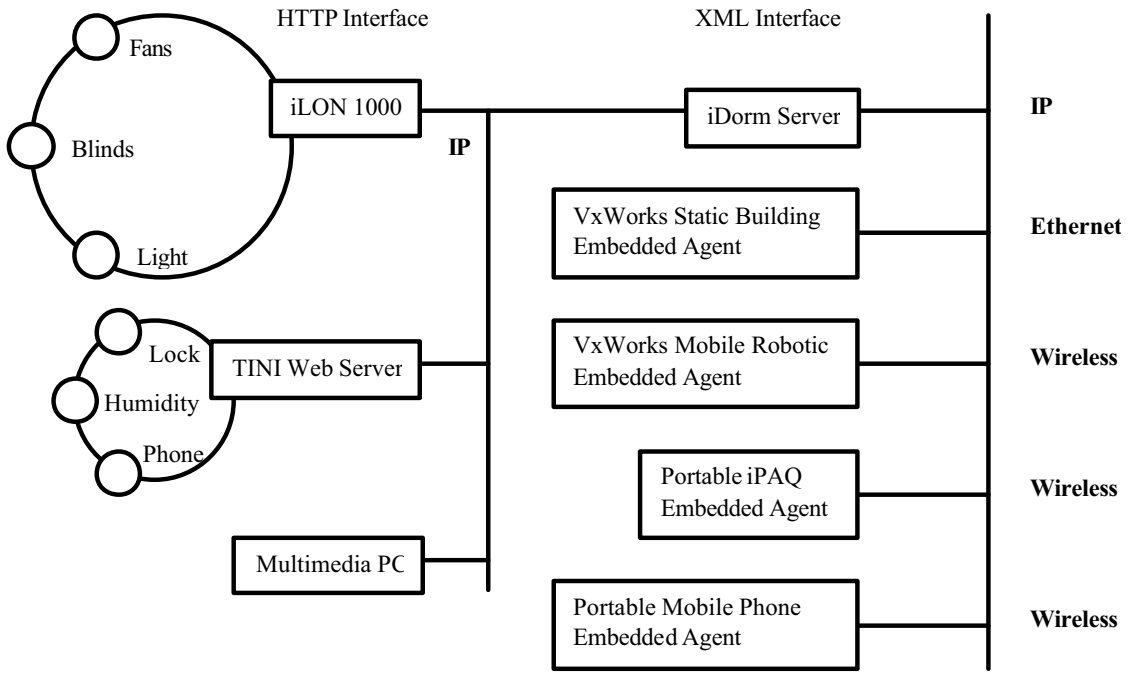


Figure (2): The logical network infrastructure in the iDorm.

The iDorm Embedded Agents

The iDorm has three types of embedded computational artefacts connected to the network infrastructure. Some of these devices contain agents.

The first type is a *physically static* computational artefact closely associated with the building. In our case this artefact contains an agent and is located in a room (a Room Embedded Agent) which receives the iDorm sensor values through the network infrastructure and contains the Incremental Synchronous Learning system to learn the user's behaviour and compute the appropriate control and send it to iDorm effectors across the network. The Room Embedded Agent (or more generically Building Agent) is shown in Figure (3-a) and is based on 68000 Motorola processor with 4 Mbytes of RAM,

an Ethernet network connection and runs VxWorks Real Time Operating System (RTOS). The sensors and actuators available to the iDorm Building Agent are as follows:

The building agent accesses eleven environmental parameters (some, such as entertainment, being parameters on multi-function appliances):

- Time of the day measured by a clock connected to the 1- Wire network
- Inside room light level measured by indoor light sensor connected to the Lonworks network
- Outside outdoor lighting level measured by an external weather station connected to the 1-Wire network
- Inside room temperature measured by sensors connected to the Lonworks and the 1-Wire networks
- Outside outdoor room temperature measured by external weather station connected the 1- wire network
- Whether the user is using his audio entertainment on the computer – sensed by custom code publishing the activity on the IP network
- Whether the user is lying or sitting on the bed or not, measured by pressure pads connected to the 1-Wire network
- Whether the user is sitting on the desk chair or not, measured by a pressure pad connected via a low power wireless connection
- Whether the window is opened or closed measured by a reed switch connected to the TINI 1-Wire network
- Whether the user is working or not, sensed by custom code publishing the activity on the IP network
- Whether the user is using video entertainment on the computer- either a TV program (via WinTV) or a DVD using the Winamp program sensed using custom code publishing the activity on the IP network

The building agent controls nine effectors, which are attached to the network infrastructure:

- Fan Heater
- Fan Cooler
- A dimmable spot light above the Door
- A dimmable spot light above the Wardrobe
- A dimmable spot light above the Computer
- A dimmable spot light above the Bed
- A Desk Lamp
- A Bedside Lamp
- Automatic blind status (i.e. open/closed and angle)

The room is also supplied with other sensors such as a smoke detector, a humidity sensor, activity sensors and telephone sensor to sense whether the phone is on or off the hook as well as a camera to be able to monitor what happens inside. It is possible to follow (and control) the activities inside the iDorm, via a live video link over the Internet.

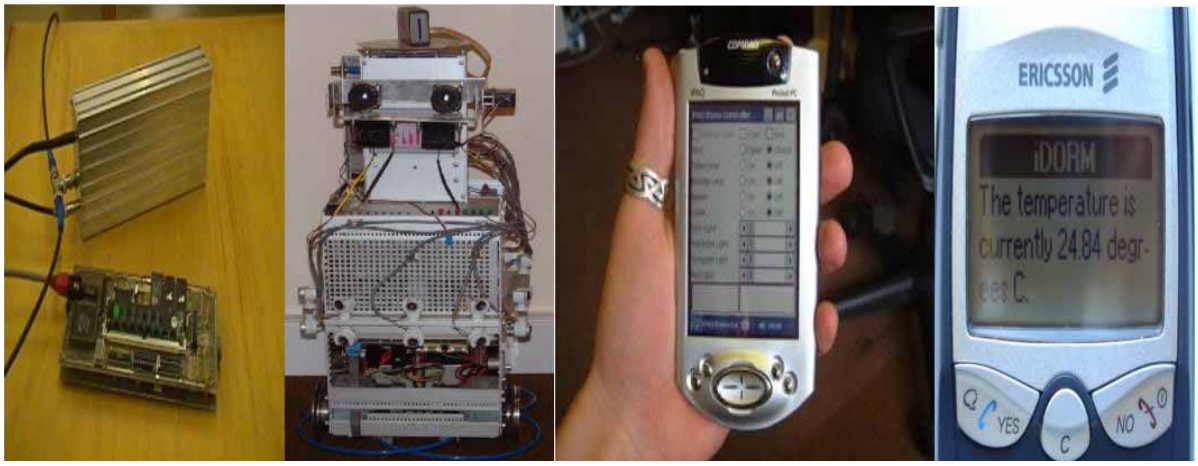
The second type is a *physically mobile* computational artefact. This takes the form of a service robot and contains an embedded agent. A prototype used in the iDorm is shown in Figure (3-b). The robot can be regarded as a servant-gadget with the aim of delivering various objects of interest to the user of the iDorm such as food, drink and medicine. The mobile robotic agent has a rich set of sensors (9 ultrasound, 2 bumpers and an IR beacon receiver) and actuators (wheels). It uses 68040 Motorola processors and runs VxWorks Real Time Operating System (RTOS). The robot is equipped with essential behaviours for navigation, such as obstacle avoidance, goal-seeking and edge-following. These behaviours are combined and co-ordinated with a fuzzy coordination module so the robot can reach a desired location whilst avoiding obstacles and is also capable of online learning of its navigation behaviour as explained in our previous work [6]. The robot's location is passed to and processed as an additional input by the *static* embedded Building Agent that controls the iDorm. In the experimental set up we use a simplified system in which the robot can go to two locations identified by infrared beacons to pick up objects. After picking up an object the robot can deliver it to the user and then go to its charging station, which is identified by another infrared beacon. The robotic agent sends information about its location to the building agent and it takes destination instructions from the building agent depending on the user's previous behaviour. For example the robot might have learned to go and fetch a newspaper from a specific location whenever it is delivered in the morning..

The communication between the *static* embedded building agent and the *mobile* robotic agent is implemented via a wireless link. Communication is established by initiating a request from the embedded building agent to the mobile embedded agent server. Once the request has been sent the server passes it to the robotic agent to carry out the task and informs the building agent of the robot's current status. If the task is in progress or not completely finished then the server sends a message indicating that the job is not complete. Every time the building agent wants to send out a new request, it waits until the previously requested job has been successfully completed.

The third type is a *physically portable* computational artefact. Typically these take the form of wearable technology that can monitor and control the iDorm wirelessly. The handheld iPAQ shown in Figure (3-c) contains a standard Java process

that can access and control the iDorm directly, this forms a type of "remote control" interface that would be particularly suitable to elderly and disabled users. Because the iPAQ supports Bluetooth wireless networking, it was possible to adjust the environment from anywhere inside and outside the room. Because the iDorm central server can also support the WML language it is possible to interact with the iDorm through mobile phones. Figure (3-d) shows the mobile phone WAP interface which is a simple extension of the web interface. It is possible for such portable devices to contain agents but this remains one of our longer term aims.

The learning mechanism within the embedded agent is designed to learn behaviours relating 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. This is mounted onto a key fob and contains a unique identification number inside its 2-kilobyte memory. The Unique ID Number of the user is passed to the embedded agent so that it may retrieve and update previous rules learnt about that user.



(a) (b) (c) (d)

Figure (3): The iDorm embedded agents a) Static Building Agent. b) Mobile Service Agent. c) Portable iPAQ interface. d) Portable mobile phone interface

Fuzzy Incremental Synchronous Learning (ISL) Technique

Broadly speaking this work is situated in the recent line of research that concentrates on the realization of artificial agents strongly coupled with the physical world. In our work, learning is done through interaction with the actual environment and we call this *online* learning 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.

The Incremental Synchronous Learning (ISL) architecture is shown in Figure (4). The ISL forms the learning engine within the embedded agent and is the subject of British patent 99-10539.7. The ISL system aims to provide life-long learning and adapts by adding, modifying or deleting rules. It is memory based in that the system can use its previous experiences (held as rules) to narrow down the search space and speed up learning. The embedded agent is an augmented behaviour based architecture, which uses a set of parallel Fuzzy Logic Controllers „ each forming a behaviour. We have used the Fuzzy Logic Controller (FLC) approach as it has proven to be very useful where the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, imprecisely or uncertainly [6], which is the case of embedded agents operating in IIE.

In general we divide the behaviours available to the embedded agent operating in the iDorm into fixed or dynamic sets, where the dynamic behaviours are learnt from the person's behaviour and the fixed behaviours are pre-programmed. These latter behaviours need to be predefined because they cannot easily be learnt e.g. the temperature at which water pipes freeze. The fixed behaviours include safety behaviour, an emergency behaviour and economy behaviour. The *Safety behaviour* ensures that the environmental conditions are always at a safe level. The *Emergency behaviour,* which in the case of a fire alarm or another emergency, might open the emergency doors and switch off the main heating and illumination systems. 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 adjustable.

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 [6, 7]. 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. The equation that maps the system input to output is given by:

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

Where M is the total number of rules, y_p is the crisp output for each rule, μ_{Aip} is the product of the membership functions for each rule's inputs and G is the number of inputs.

We use a higher level FLC to combine the preferences of different behaviours into a collective preference (giving a two-level behaviour hierarchy). In this model, command fusion is decomposed into two steps: preference combination and decision making and in the case of using fuzzy numbers for preferences, product-sum combination and height defuzzification.

The final output equation is given by:

$$Y_{ht} = \frac{\sum_{i} (mm_y * y_t)}{\sum_{i} mm_y} \qquad (2)$$

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 dynamically taking into account the context of the agent.

The output of each FLC is then fed to the actuators via the *Co-ordinator* that weights its effect. More information about the fuzzy hierarchical architecture can be found in [6, 7].

The ISL works as follows: - when a new user enters the room they are identified by the active key button and the ISL enters a Monitoring initialisation mode where it learns the user's 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 completely we want the initial rule base. For example in a care home we might want this rule base to be as complete as possible, in a hotel we might 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 reactivated whenever the user renters the room. During this initialisation period the system monitors the inputs and the user's action and tries to infer rules from the user's behaviour. The user will usually act when a set of environmental conditions (an input vector) is unsatisfactory to him by altering the output vector (he needs to turn a light on or adjust the heating etc). 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 room's effectors. Whenever the user behaviour changes, there may be a need to modify, add or delete some of the rules in the rule base. Thus the ISL goes back to the non intrusive cycle and tries to infer the rule base change to determine the user's preferences in relation to the specific components of the rule that has failed. This is a very short cycle that the user is essentially unaware of and such modifications are distributed throughout the lifetime of the use of environment, thus forming a life-long learning phase.

As in the case of classifier systems, in order to preserve 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 *m* new classifiers [5]. In our case we will change all the consequents of the rules whose consequents were unsatisfactory to the user. We find these rules by finding all the rules firing at this situation whose firing strength $\mu Ai > 0$. We replace these rule consequents by the fuzzy set that has the highest membership of the output membership function. We make this replacement to achieve non-intrusive learning avoiding 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.

During the non-intrusive monitoring and life-long learning phases the agent encounters many different situations as both the environment and the user's behaviour change, e.g. the agent attempts to discover the rules needed in each situation guided by the occupant's behaviour in response to different temperature and lighting levels inside and outside the room.. 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 [3]. 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 complete rule base all at once but we learn only the rules needed by the user during the different episodes. There is 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.

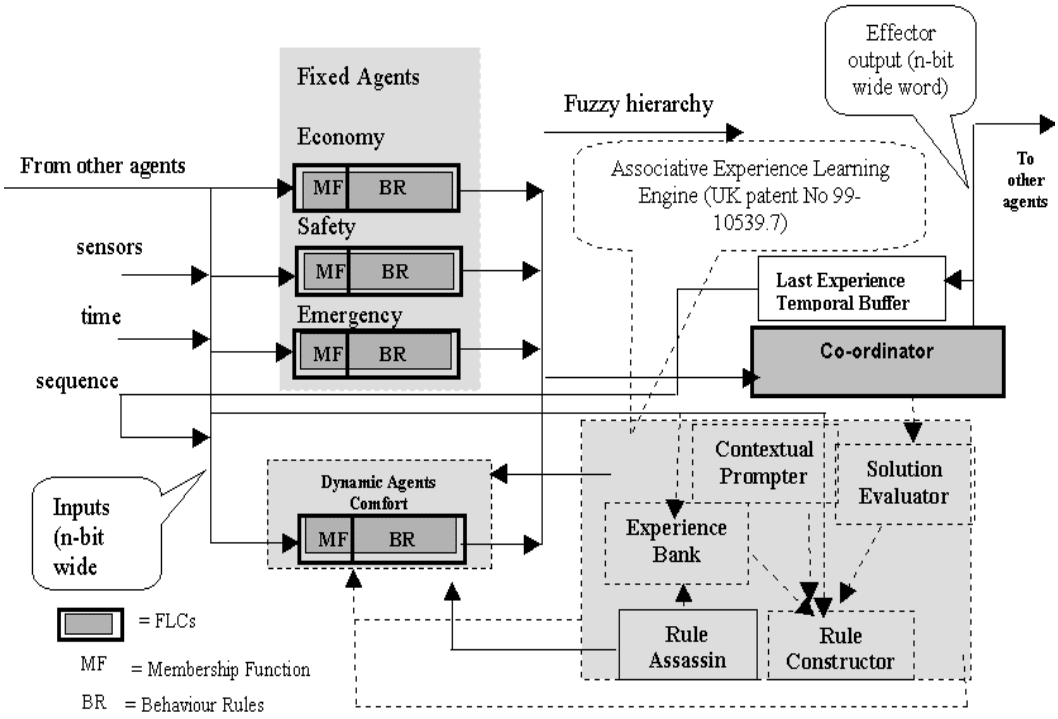


Figure (4): The ISL Embedded-Agent Architecture

After the initial monitoring phase the system 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. By doing this the system is trying to predict the rules that were not fired in the initialisation session thus minimising the learning time as the search starts from the closest rule base rather than starting from a random position. This action should be satisfactory for the user as the system starts from a similar rule-base then fine-tunes the rules.

Subsequently the agent will operate with rules learnt during the monitoring session plus rules that deal with situations uncovered during the monitoring process, which are ported from the rule base of the most similar user, all the rules that are constructed and added to the system are symbolised by the *Rule Constructor* block in Figure(4). 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 briefly re-entering the Monitoring mode. In this case again the system finds the rules fired and changes their consequent to the action exhibited by the user. 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 user's instruction unless safety is compromised. 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, as we are dealing with embedded agents with limited computational and memory capabilities, it is very difficult to deal with a large number of rules in the rule base, e.g. for the comfort behaviours the complete set of possible rules for the iDorm is 62208 rules and this would 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 since it was added to the system. In calculating this *degree of importance* we also include a measure of *most-recent-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. In order not to lose the rules that are chosen for replacement we 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 frequently used rules and not degrade the real time performance of the embedded agent.

Experimental Results

We have conducted a novel experiment over five and a half days (132 hours) in which a user occupied the iDorm while the iDorm was under the control of the building embedded agent using the ISL architecture.

Figure (5): a) The Agent Communication Path. b) The user is sleeping in the bed in the iDorm, which is controlled by the ISL. c) Using the active lock. d) Another User is sitting at the desk in the afternoon in the iDorm, which is controlled by the ISL

The user was identified by his intelligent key, which activates the active lock as shown in Fig (5-b). Throughout the experiment, the user used the wireless iPAQ to monitor and control the iDorm environment 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 changes to controls occurred, the building embedded agent received the request, generated a new rule or adjusted a previously learnt rule and allowed the action through as shown in Figure (5-a). A small parsing tool was written to convert the text file containing the fuzzy rule sets 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 intending. The second was to compare the number of rules learnt over time. The embedded agent's success can be measured by monitoring how well it matches the environment to the user's demands. If it does this well, the user will intervene less and

cause less rule generation over time. If it does this badly, the user will intervene more and cause more rule generation over time.

The results of the experiment are summarised in Figure (6). The data making up the graph was generated by a monitoring program which stored the number of rules learnt by the ISL along with a time stamp. It took a reading every five minutes for the duration of the experiment. There are several sections of the graph worthy of note.

In the first section from *0-24 hours* starting from an initial rule base of zero, it can be seen that a very large number of rules are learnt in a comparatively short period of time. In fact, in the first nine hours of the experiment, the agent learnt 128 rules. This is nearly half the total number of rules learnt by the end of the experiment. These results are consistent with the agent learning and making incorrect decisions for the user in the initial stage. However, as the end of the period is reached it can be seen that the learning rate of the agent (Rules/Time) reduces drastically. This is consistent with the agent's reactions requiring less correction by the user and is therefore consistent with the agent making useful decisions about the environment state based on the user's requirements. The latter trend of fewer rules learnt over time is consistent across the whole of the experiment. Hence the level of "comfort" experienced by the user (in relation to the environment state) is high enough for them not to make an environmental change and consequently alter the learning rate.

The second section of the graph from *60-72 hours* shows a sharp increase in the learning rate of the agent. This is explained by the user introducing novel activity into his repertoire of behaviours as the system operates in a life long learning mode.

The third section 72-132 hours shows that in last two days the agent had not generated any new rules. This constitutes a control period where the user was generally satisfied with agent control action that resulted from the embedded agent successfully learning the user behaviour.

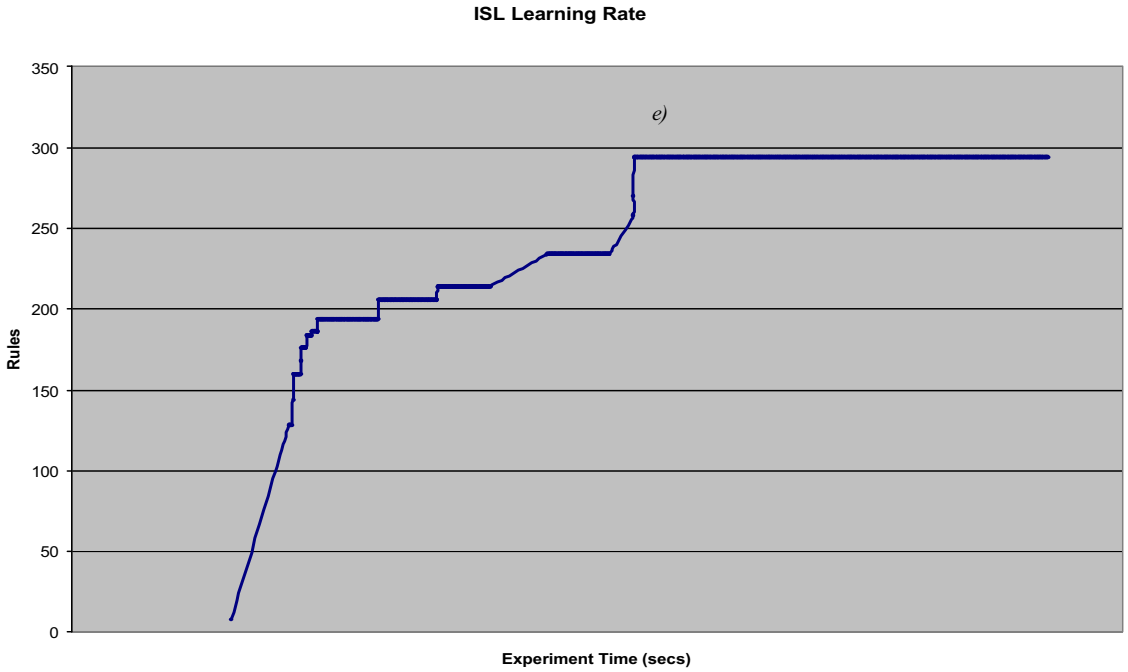


Figure (6): The rules learnt plotted against experiment time

The agent has learnt the 280 rules needed to capture the behaviour of this user over the 132 hours experiment, which demonstrated that our system can learn effectively using the ISL and it doesn't need to learn the complete rule base (62208 rules in case of the iDorm). Figure (6) shows that the agent had to learn less new rules about the user as the experiment progressed. Since this was one of our criteria for measuring the agent's success the evidence of the continual reduction in the learning rate, leads us to conclude that the agent managed to pick out the pertinent behaviour of the user over time.

An example of a rule learnt by the agent could be written as follows:

> *In the evening when it is dark inside and bright outside, temperate inside and warm outside, sitting on the bed, with the window open I switch the heater off, switch the cooler off, switch the door spotlight off, switch the wardrobe spotlight off, switch the computer spotlight off, switch the bed spotlight off, turn off the bed lamp turn on the table lamp and close the blind and send the robot to the bed.*

The rules involves the robot moving to the user (who is lying on the bed) at specific input vector. The mobile robot was used as a method of transporting specific objects to the user such as a pen or a can of drink. These objects were placed on a flat area of the robot that could be used as a tray, retrieved, used and replaced when finished.

In our previous work we tried experimenting with different room users [6] and the role of the Experience Bank was important in reducing the time it takes to learn the user's behaviour and achieve user satisfaction. This is because it starts learning the user's behaviour from the best matching behaviour previously recorded rather than starting from random.

These experiments showed how our embedded agent approach can realise the vision of ambient intelligence in a ubiquitous computing test bed environment like the iDorm by learning the user behaviour and controlling the iDorm accoriding to the user desires. The learning was done in a non intrusive way without the user aware of the existanmce of the agent.

Conclusions

Using our embedded agent approach in the iDorm we have demonstrated the vision of ambient intelligence in ubiquitous computing environments, where our emebeded the agent was able to learn the user behaviour and control the iDorm without the user being aware of the agent. We argue that embedded-intelligence can bring significant cost and effort savings over the evolving lifetime of products 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 cognitive overhead.

We have performed a 5 day experiment using the iDorm and demonstrated that our fuzzy logic based ISL can develop useful particular rules over a short time frame. This technique has the ability to add, delete or modify rules in the dynamic behaviour rule base which is an essential feature of an "always oN" life-long learning embedded agent. We have shown that the ISL agent is both immediately reactive to a user's command (subject to safety restrictions) and is able to particularise itself to the user's behaviour (including idiosyncratic actions) rather than by generalising for a group of users, both of which are an essential feature of any agent that is to be acceptable to people. These experiments suggest that surprisingly few rules are required by the agent (i.e. in the order of a few hundred) in order to autonomously create a comfortable environment with diminishing need for user correction.

We believe that these results support the idea that human behaviour can be usefully modelled without high processing power or large amounts of memory. Indeed, the results suggest that over the experimental period the agent made a significant reduction in the cognitive load of the user. The shape of the learning curve in Figure (6) also suggests that the agent moved increasingly closer towards the user's environmental preference even though this preference was never static.

Future Work

Our current experimental programme includes plans for multi-user habitation experiments and wider deployment of embedded agents (e.g. personal agents inside wearable technology). We are also in the process of building a multi-roomed version of the iDorm, iDorm-2, as a preliminary step towards the construction of a fully functional apartment (iFlat) which will house both visiting researchers and act as a pervasive computer test-bed. The iFlat is currently being designed for such experimentation from the ground up.

Acknowledgements: We are pleased to acknowledge the funding support from the EU IST Disappearing Computer program (eGadgets) and the Korean-UK Scientific Fund programme (careAgents).

References

[1] G. Abowd, M. Eibling, G. Hunt G, H. Lei, "Context Aware Computing" IEEE Pervasive Computing, Volume 1, No 3, ,pp.22-23, 2002

[2] R. Brooks, "Intelligent Room Project", Proc 2nd Int'l Cognitive Technology Conference (CT'97), Japan 1997.

[3] V. Callaghan, M. Colley, G. Clarke, H. Hagras, "Embedding Intelligence: Research Issues for Ubiquitous Computing", Proceedings of the Ubiquitous Computing in Domestic Environments conference, Nottingham, September 13 - 14, 2001.

[4] P. Davidsson "Energy Saving and Value Added Services; Controlling Intelligent-Buildings Using a Multi-Agent System Approach" in DA/DSM Europe DistribuTECH, PennWell, 1998.

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

[6] H. Hagras, V. Callaghan, M. Colley, G. Clarke, H. Duman, "Online Learning and Adaptation for Intelligent Embedded Agents Operating in Domestic Environments " In the book entitled Fusion of Soft Computing and Hard Computing for Autonomous Robotic Systems" (Eds.: Zhou, Dario Maravall and Da Ruan), "Studies in Fuzziness and Soft Computing Series , Physica-Verlag, Volume 116, pp. 293-323,November 2002.

[7] H. Hagras, V. Callaghan, M. Colley, G. Clarke, "A Hierarchical Fuzzy Genetic Multi-Agent Architecture for Intelligent Buildings Learning, Adaptation and Control", The International Journal of Information Sciences, *Volume. 150, pp. 33-54, March 2003*..

[8] A. Holmes, H. Duman, A. Pounds-Cornish "The iDorm: Gateway to Heterogeneous Networking Environments", International ITEA Workshop on Virtual Home Environments, 20-21st February 2002. Paderborn, Germany.

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

[10] A Sherwin. Internet Home Offers a Life of Virtual Luxury. The Times, pp. 10, 3rd Nov 1999.

[11] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, July 1993. (reprinted as "Ubiquitous Computing". *Nikkei Electronics*; December 6, 1993; pp. 137-143.)

