In the International Series "Studies in Fuzziness and Soft Computing",
Physica Verlag, Volume 116, pp. 293-323, November 2002

# Online Learning and Adaptation for Intelligent Embedded Agents Operating in Domestic Environments

Hani Hagras, Victor Callaghan, Martin Colley, Graham Clarke, Hakan Duman

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

## Abstract

In this paper we show how intelligent embedded agents situated in an intelligent domestic environment can perform learning and adaptation. A typical domestic environment provides an environment where there is wide scope for utilising computer-based products to enhance living conditions. Intelligent embedded agents can be part of the building infrastructure and static in nature (e.g. lighting, HVAC etc.), some will be carried on the person as wearables, others will be nomadic in nature remaining static for periods but occasionally moving between locations or highly mobile, as with robots. Both non-intrusive and interactive learning modes (including a mix of both) are used, depending on situation of the agent. For instance mobile robot type agents use an interactive learning whilst most building based agents use non-intrusive background learning modes. In this paper we will introduce the learning and adaptation mechanisms needed by the Building and Robotic embedded agents to fulfil their missions in intelligent domestic environments. We also present a high-level multi embedded-agent model, explaining how it facilitates inter-agent communication and cooperation between heterogeneous sets of embedded agents within a domestic environment.

## 1. Introduction

The variety of computer-based goods, and their capabilities, is growing at an unprecedented rate fuelled by advances in microelectronics and Internet technology. Cheap and compact microelectronics means most everyday artifacts (e.g. shoes, cups) are now potential targets of embedded-computers, while ever-pervasive networks will allow such artifacts to be associated together in both familiar and novel arrangements to make highly personalized systems.

A typical domestic environment provides an environment where there is wide scope for utilizing computer-based products to enhance living conditions. For instance it is possible to automate building services (e.g. lighting, heating etc), make use of computer based entertainment's systems (e.g. DVDs, TV etc), install work tools (e.g. robot vacuum cleaners, washing machines, cookers etc), or enhance peoples safety (e.g. security and emergency measures, appliance monitors etc). Some of these artifacts will be part of the building infrastructure and static in nature (e.g. lighting, HVAC etc.), others will be carried on the person as wearables or mobiles, or temporarily installed by people as they decorate their personal space (e.g. mobile phones, TVs etc).

In order to realise the intelligent domestic environments, technologies must be developed that will support ad-hoc and highly dynamic (re) structuring of such artifacts whilst shielding non-technical users from the need to understand or work directly with the technology "hidden" inside such artifacts or systems of artifacts. For this vision to be realized in domestic environments, people must be able to use computer-based artifacts 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 as, for example, with a video recorder, whose 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 artifacts increase. We argue that if some part of the reasoning, planning and learning normally provided by a gadget user, were embedded into the artifact 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). However, this is far from easy as such "intelligent artifacts" operate in a computationally complex and challenging physical unstructured environment which is significantly different to that

encountered in more traditional PC programming or AI. A major challenge is the large amount of uncertainty that characterizes real world environments. On the one hand, it is not possible to have exact and complete prior knowledge of these environments: many details are usually unknown. On the other hand, knowledge acquired through sensing is affected by uncertainty and imprecision. The quality of sensor information is influenced by sensor noise, the limited field of view, the conditions of observation, and the inherent difficulty of the perceptual interpretation process.

In this chapter, we describe an innovative multi heterogeneous agent environment consisting of a domestic environment inhabited by a variety of agents. Intelligent embedded agents can be part of the building infrastructure and static in nature (e.g. lighting, HVAC etc.), some will be carried on the person as wearables or mobiles (termed PA), others will be mobile robotic agents (termed RA). The class of agents we term Building Agents (BA) are situated in the building services and try to learn the occupant's habitual behaviour and preemptively adjust the environment to satisfy him via a non-intrusive learning mode. Intelligent Robotic Agents differ in that they learn behaviours through interaction with the environment. An essential feature that characterizes all our work is that intelligent habitat technology needs to be centered on the individual, tailoring themselves to an individual wherever possible, rather than generalizing across a group of individuals.

In the next section we will introduce intelligent autonomous embedded agents and explain firstly why it is important to learn online and secondly the methods used for learning. We then introduce our heterogeneous multi agent architecture for intelligent domestic environments that features a hierarchical fuzzy genetic system for online learning and adaptation. We describe interactive learning in the mobile Robotic Agents and non-intrusive learning in the Building Agents. Finally we offer experimental results, our findings to date and plans for future work.

### 1.1 Intelligent Autonomous Embedded Agents

Embedded intelligence can be regarded as the inclusion of some of the reasoning, planning and learning processes in an artifact that, if a person did it, we would regard as requiring intelligence. An intelligent artifact would normally contain only a minimal amount of "embedded-intelligence", sufficient to do the artifact task in question. Embedded-computers that contain such an intelligent capability are normally referred to as "*embedded-agents*" (Callaghan et al. 2000). Intelligent Artifacts would, in effect, contain an embedded-agent. Individually, such an embedded-agent can harness intelligence to undertake such tasks as:

- Enhancing Artifact functionality (enabling the artifact to do more complex tasks)
- Simplifying or automating the user interface (in effect, providing an intelligent assistant)
- Reducing Programming Costs (the system learns its own program rules)

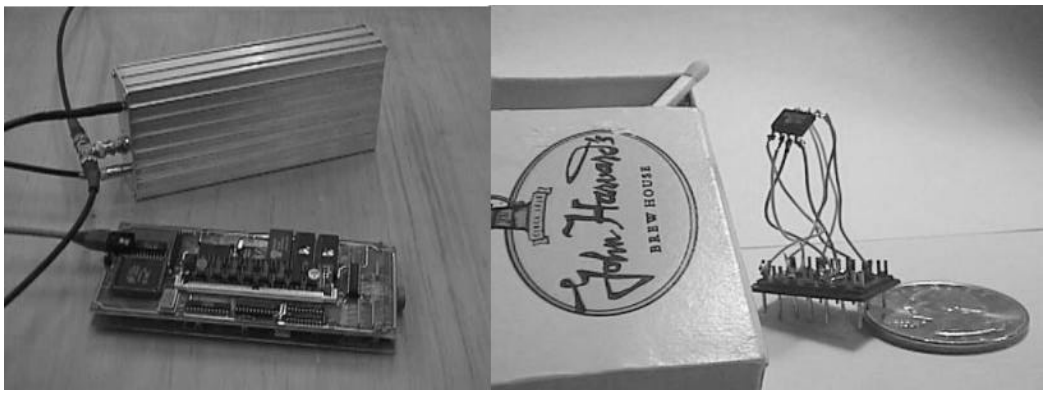
It is now common for such "*embedded-agents*" to have an Internet connection thereby facilitating multi embedded-agent systems. In a fully distributed multi embedded-agent systems each agent is an autonomous entity co-operating, by means of either structured or ad-hoc associations with its neighbors. Each agent can reason or plan how it might work with those with which it is currently associated thereby supporting *evolving aims or emerging functionality (cf system customization).* It is important to understand that being autonomous and promiscuous (open to making associations with other artifacts) does not imply undirected or unsafe behavior. Agents can have basic fixed rules built in to them that prevent them taking specified actions deemed unsafe.

Most automation systems (which involve a minimum of intelligence) utilize mechanisms that generalize actions (e.g. set temperature or volume that is the average of many people's needs). However, we contend that AI applied to personal artifacts and spaces needs to *particularize* itself to the individual [Callaghan 2001]. Further, subject to safety constraints, we contend that it is essential that any agent serving a person should always and immediately carry out any requested action, no matter how perverse it may appear (i.e. people are always in control, subject to overriding safety considerations). The embedded-agent techniques we will outline are characterized by their ability to particularize their actions to individuals *and* immediately execute commands, wherever that is a practical possibility. Thus, the value of an intelligent autonomous 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 [Callaghan 2001]. 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 there is a need for 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.

According to Kasabov [Kasabov 98] an Intelligent Agent System (IAS) should be able to learn quickly from large amounts of data. He also states that an intelligent system should also adapt in a real time and in an on-line mode as new data is encountered. The system should also 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 analyze itself in terms of behaviour, error and success. To our knowledge, no system in the field of embedded agents operating in unstructured environments had satisfied these criteria (Hagras et al 2001a).

For physical disappearance artifacts will need relatively small low-cost embedded computers (possibly based on application specific micro-electronic fabrication). For example typical specifications might be Cost: £20-£50, Size: $<2^2$cm, Speed: 1-10MHz, Memory: 1-2 MB, I/O: 10-50 I/O channels. Examples of two real devices are shown in figures 1(a) & 1(b).




(a) (b)

Figure (1): a) University of Essex Prototype building services agent. b) University of Massachusetts Prototype Embedded-Internet Device

While it is inevitable that the “computing power / cost ratio” will continue to increase (i.e. more mega-everything per dollar), history has shown that functionality will always demand even faster computers. Thus available resources for a given cost always lag behind needs. The classic illustration of this dilemma is the defiance of the hard disk to become extinct despite 30 years of predictions of semiconductor memory becoming cheap and abundant. Of course the prediction that memory will become cheap and abundant has always proved correct but it seems functional demands have outpaced it. The lesson here is that although it is inevitable that embedded-computers will become much more powerful, they will always be less powerful than the functionality demanded at that future point!

Traditional artificial intelligence (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 2 orders of magnitude removed from most embedded-computers. In addition traditional AI techniques have proved too fragile to operate real time intelligent machines such as robots. As a result, even implementing simplified traditional AI systems on embedded-computers has proved virtually impossible. However, the authors have techniques from developments of their earlier work in robotics that seem well suited to providing artifact intelligence (Callaghan et al. 2001, Hagras et al. 2001a, Hagras et al. 2002) which are discussed later in this chapter.

### 1.2 Why Online Learning

Broadly speaking this work situates itself in the recent line of research that concentrates on the realization 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 according to the above definition of embedded agents. Another important point is that adaptive behavior 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 95). 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 criticized and the subject of much debate. Even so, computer simulations may still be very helpful in the training and testing of agent models. However as Brooks (Brooks 92) 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. There are several reasons why those using computer models (simulations) to develop control systems for embedded agents operating in unstructured and changing environments may encounter problems (Miglino et al. 95) as numerical simulations do not usually consider all the physical laws of the interaction of a real agent with its own environment, such as mass, weight, friction, inertia, etc. Also physical sensors deliver uncertain values, and commands to actuators have uncertain effects, whereas simulative models often use grid-worlds and sensors that return perfect information. Physical sensors and actuators, even if apparently identical, may perform differently because of slight variations in the electronics and mechanics or because of their different positions or because of the changing weather or environmental conditions.

Even where researchers are using real embedded agents in the real world to learn behaviors, these behaviors if learnt successfully are usually frozen within the agent. Thus if some of the agent dynamics or the environmental circumstances are changed, the agent must repeat a time-consuming learning cycle (Miglino et al. 95). From the above discussion it is clear that using computer simulations for developing agent controllers has significant disadvantages which is best illustrated by the fact that when transferring the learnt controllers from the simulated world to real world these controllers will usually fail [Miglino et al 95]

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.

### 1.3 Other work in developing Intelligent Embedded Agents for Intelligent Inhabited Spaces

There are a growing number of research projects concerned with applying Artificial Intelligence (AI) to intelligent inhabited spaces. In Sweden, Davidsson (Davidsson 98) utilised 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.. In Colorado (Mozer 98) used neural networks to Their system, implemented in a building with a real occupant, also achieved a significant energy reduction. Work at MIT on the HAL project concentrates on making the room responsive to the occupant by adding intelligent sensors to the user interface (Brooks 97). The University of Loughborough (Angelov et al. 2000) looked at the application of fuzzy rule-based models in HVAC system simulation. This project is concerned with producing optimal models for buildings, which could be used later in control. The HIVE project at MIT (Minar et al. 99) is an example of a particularly forward-looking distributed agent model. This model differs from our work principally in respect 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. 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 (Sherwin 99). 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 had addressed online learning and adaptation of a hertogeneous set of agents within a domestic environment.

Adaptive Soft Computing Approaches?

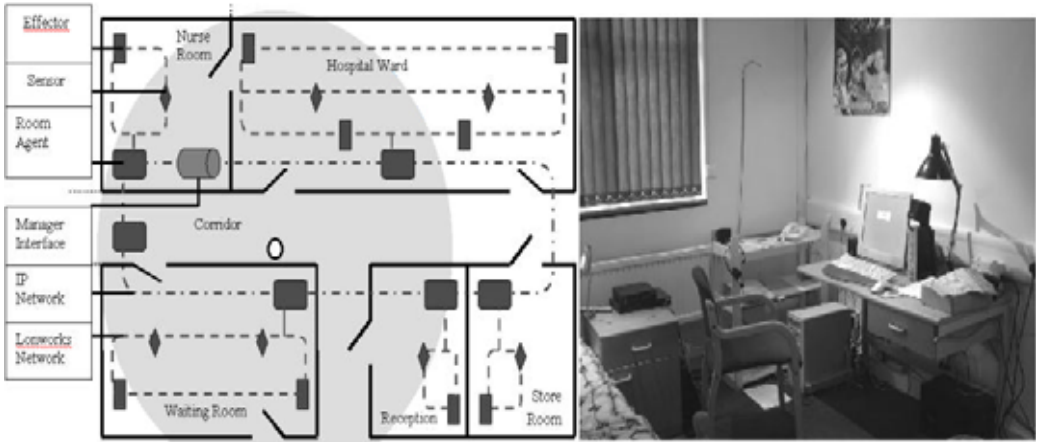
The methodology of Fuzzy Logic Control (FLC) appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, imprecisely or uncertainly [Pedrycz 98], which is the case of autonomous embedded agents. However in complex unstructured environments the necessity of high quality information, to be extracted from a broad knowledge domain, constrains the application of fuzzy systems to solve very demanding problems. Also as the number of inputs variables increases (which is the case of embedded agents) the number of rules increases exponentially which creates much difficulty in determining large numbers of rules.

Evolutionary algorithms constitute a class of search and optimisation methods guided by the principles of natural evolution and genetics. It is the case that Genetic Algorithms (GA) have been successfully applied to solve a variety of difficult theoretical and practical problems by imitating the underlying processes of evolution such as selection, recombination and mutation. GA are implicitly parallel where the solution is explored in parallel by searching different regions, this characteristic allows a global search in the solution space (Kasabov 98). GA approach enriches the optimisation environment for fuzzy systems. As described in (Delgado et al. 2000] developing an optimal fuzzy system is equivalent to finding the minimum of a hyper-surface associated with an objective function. The hyper-surface has the following characteristics: it is infinitely large, complex and noisy; and is non-differentiable. Since changes in the number of fuzzy rules are discrete and can have a discontinuous effect on the fuzzy system's performance; it is multi-modal (different fuzzy rule sets and/or membership functions may have similar performance). It is also deceptive, since a little modification may cause huge effects on the performance of each system (Delgado et al. 2000]. There is much work reported in the literature on designing fuzzy controllers using GA (Hoffmann 98, Matellan et al. 98, Bonarini 99, Delgado et al. 2000, Sousa and Madrid 2000). However virtually most of this work was undertaken using simulation as, in conventional GA, it takes a large number of iterations to develop a good controller. Even when researchers are using real embedded agents to learn behaviours online, these behaviours if learnt successfully are usually frozen in the agents. Thus if some of the robot dynamics or the environmental circumstances are changed, the robot must repeat a time-consuming learning cycle (Miglino et al. 95]. Thus it is not feasible for a simple GA to learn online and adapt in real-time. The situation is worsened by the fact that most evolutionary computation methods developed so far assume that the solution space is fixed (i.e. the evolution takes place within a pre-defined problem space and not in a dynamically changing and open one), thus preventing them from being used in real-time applications (Kasabov 98). Hence prior to our work it was *not considered feasible for a simple GA to online learn and adapt an embedded agent controller* (Linkens and Nyongeso 95) in unstructured domestic environments.

## 2. Heterogeneous Multi-Agent Domestic Intelligent Environments Application

Figure (2-a) presents an intelligent domestic environment for care/rehabilitation system in which a collection of building and robotic agents cooperates to care for human occupants (Colley et al. 2001). The BA handles the control of all building services (e.g. heat, light entertainment, etc) by being attached to various sensors and controllers. RA communicate with BA so as to operate more effectively (e.g. command doors to open, locate themselves, responding to remote requests, etc) providing services such as delivery of meals, medicine etc. Wearables allow remote monitoring of the patient and can be set to signal any worrying deviation from expected values.

We have chosen the Essex Intelligent Dormitory (iDorm) shown in Figure (2-b) to form the experimental framework for the domestic 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 room for elderly or disabled people or an intelligent hotel room. Because this room is of an experimental nature we are fitting it with a liberal placement of sensors (e.g. temp. sensors, presence detectors, system monitors etc) and effectors (e.g. door actuators, equipment switches etc), which the occupant can configure and use. The room looks like any other but above the ceiling and behind the walls hides a multitude of networks and networked devices.




(a) (b)

Figure (2): a) Example Intelligent Environment Infrastructure. b) Photograph of the iDorm

The iDorm is based around three networks, Lontalk, Tini 1-wire and IP. This provides a diverse infrastructure and allows the development of network independent solutions. It also gives us an opportunity to evaluate the merits of each network.

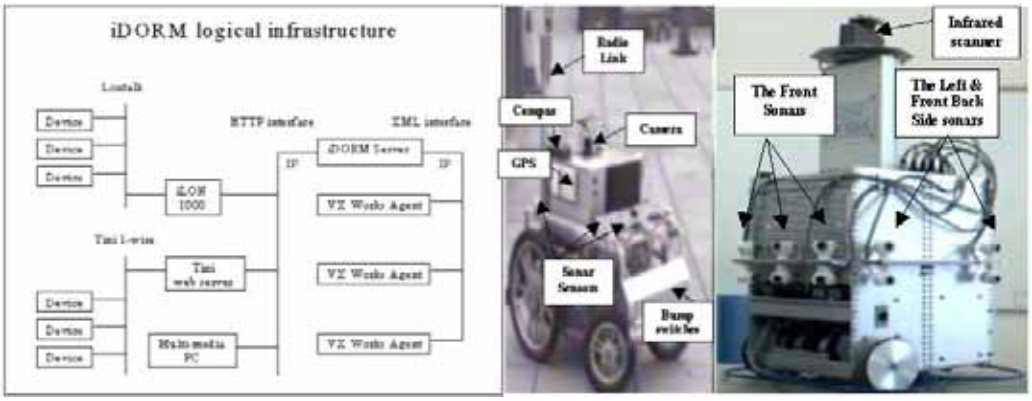
To create a standard interface to the iDorm we have an iDorm gateway server. This exchanges XML formatted queries with the entire principal computing components, which overcomes many of the practical problems of mixing networks. The communications architecture is being extended to allow devices to be 'Plug N Play' (enabling automatic discovery and configuration). The iDorm logical infrastructure is shown in Figure (3-a).

The embedded agent used for the intelligent buildings shown in Figure (1-a) is based on a 68000 Motorola processor with 4 Mbytes of RAM and an Ethernet network connection. It runs the VxWorks Real Time Operating System (RTOS).

Our mobile robots are based around a distributed field bus control system. In particular, we use the CANbus (Controller Area Network) developed for automotive industry, Motorola processors and the VxWorks Real Time Operating System (RTOS). The current design is influenced largely by the requirements for both parallel and distributed processing in a real-time environment. We will use different sizes of robots for our experiments to verify that it is robot independent. We will also perform the robot experiments in difficult outdoor unstructured environments to test online learning and adaptation. The outdoor robot is electric. We will use ultrasound for outdoor navigation, as it has proved convenient and well able to cope with outdoor environments. The ultrasound transceivers were designed in a compact unit with the transmitter and receiver in the same unit and all the analogue components are fine-tuned for a perfect operation. It was also equipped with amplification circuits to try to pick even the weak reflected signals. It has also extra filtering and noise immune circuits to deal with the noise present in outdoor. The robot is also supplied with GPS, compass and a camera for goal determination in outdoor environments. We try to give all our robots a similar architecture (to simplify development work) so its hardware is also based on embedded Motorola processors (68040) running VxWorks RTOS. The outdoor robot has two different motors one for controlling the speed of the front wheels and the other for controlling the steering of the front wheel. The control programs are developed under the Tornado environment and then downloaded via an Ethernet cable or using RF modem to the robots. After the program downloading the cable or the RF link can be disconnected and the navigation becomes autonomous. The electrical robot and its sensor configuration are shown in Figure (3-b). The indoor robot has a ring of 7 ultrasonic proximity detectors, an 8-axis vectored bump switch, an IR scanner sensor (to aid navigation) and two independent stepper motors for driving plus

differential steering. The hardware is based on embedded Motorola 68040 processors running the VxWorks RTOS. In the initial experiments infrared beacons were used to simulate the goals in indoor environments. Control programs are developed using the Tornado VxWorks environment and then downloaded via an ethernet cable to the robots, after which the cable can be disconnected and the navigation is autonomous. The robot and its sensor configuration are shown in Figure (3-c).

The RA can also take the form of a Manus robot arm, which can even be located at a remote place. We are currently involved in a collaborative project with the Korea Advanced Institute of Science and Technology (KAIST) supported by UK-Korea S&T collaboration fund. In this project our BA in the University of Essex, UK will be cooperating and communicating with RA located in Essex and also remotely to RA in Korea.




(a) (b) (c)

Figure (3): a) iDorm logical infrastructure. b) The outdoor electrical robot and sensors. c) The indoor robot and sensors

Currently the communication between the BA and the RA is established by initiating a request command from the BA to the RA server. The server creates the link between the computers and waits for incoming commands from the BA. Depending on the current situation in the building environment the BA sends out commands to the RA such as moving the Manus Arm to a certain position or commanding the mobile robot to pick up the mail. Once the command has been sent the server passes the request to the responsible RA to fulfill the task and informs the 'commander' (Building agent) about the current status of the robots. If the task is not complete then the server sends a message indicating that the job is "not complete". Every time when the BA wants to send out a new command, it waits until the previously requested job has been successfully finished.

Internet-based control systems rely on the available communication protocols to exchange real-time data between two computers. Most network protocols nowadays provide a reliable and transparent support for data exchange among computers by using protocols such as the Transmission Control Protocol (TCP). Real-time control is used in systems that must react to external stimuli with minimal delay in order to maintain stability. The issue of time delay is not the main subject in this chapter but it has been addressed by applying a feedback system. The designed system depicted below allows the RA (Manus arm and mobile robot) to continuously execute new coming commands while transmitting feedback information on if the job has been completed or not.

The communication between the BA and the RA are implemented by applying a TCP/IP stream socket. The BA has several I/O interfaces. One of them is used to connect to an IP network. To establish the communication the Agent uses *Stream Sockets* to communicate with a TCP port within the node. In other words *Stream sockets* use TCP to bind to a particular port number. Another process, on any host in the network, can then create another stream socket and request that it be connected to the first socket by specifying its host Internet address and port number. After the two TCP sockets are connected, there is a *virtual circuit* set up between them, allowing reliable socket-to-socket communications.

Streaming data transfer means that applications using TCP do not have to break up data into blocks before sending them. Once data is passed to TCP, it transmits the data in a stream of bytes that are tagged with sequence numbers. These numbers are checked when they arrive at the remote computer and if a packet in the sequence is lost, a time-out mechanism requests retransmission.

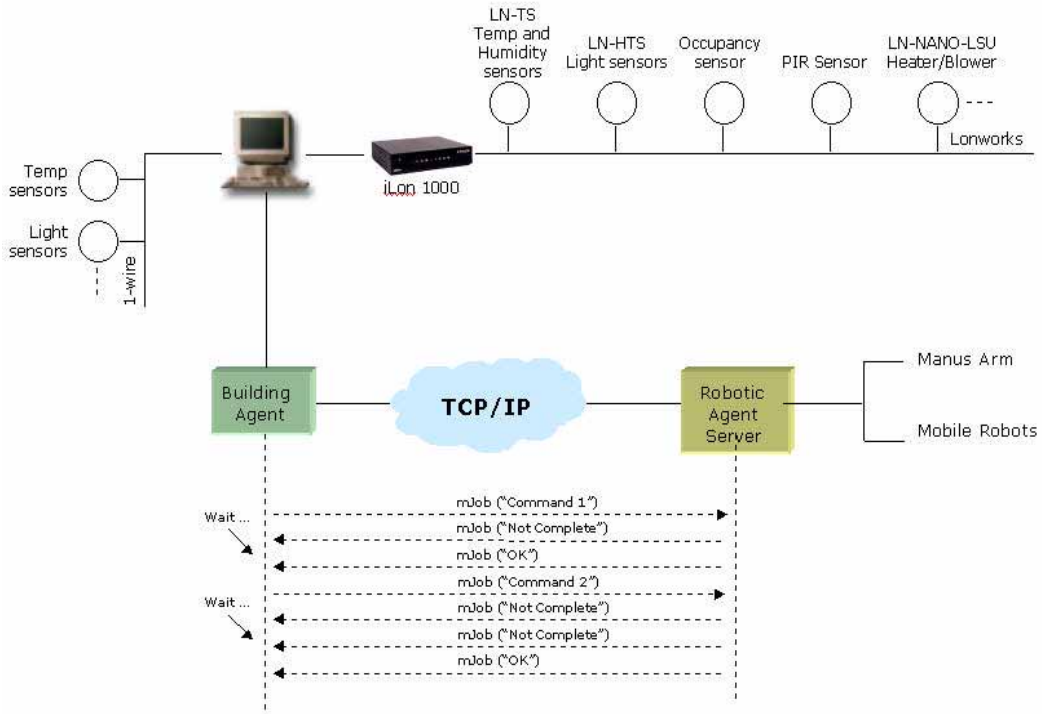



Figure (4): Framework of the heterogeneous Multi Embedded Agents communication

## 3. Hierarchical Fuzzy Logic Controllers (HFLC)

Most commercial Fuzzy Logic Control (FLC) implementations feature a single layer of inferencing between two or three inputs and one or two outputs. For autonomous embedded agents, however the number of inputs and outputs are usually large and the desired control behaviours are more complex. In the robots domain we use eight inputs (7 sonar inputs and a bearing sensor) and two outputs (the speed and the steering). If we assume that each input will be represented by three fuzzy sets and each output by four fuzzy sets, using a single layer of inferencing will lead to determining $3^8 = 6561$ rules which would be difficult, if not impossible, to determine. However, by using a *hierarchical* assembly of fuzzy controllers, the number of rules required can be significantly reduced. For robot navigation the experimental system can be divided into four co-operating behaviours, obstacle avoidance, left and right wall following and goal seeking. If these behaviours represent each input by three fuzzy sets the obstacle avoidance with three inputs produces $3^3 = 27$ rules. The left wall following has two inputs producing $3^2 = 9$ rules, right wall following is the same and goal seeking has one input (more accurately represented by seven fuzzy sets) producing 7 rules. Thus the total number of rules now required is $27 + 9 + 9 + 7 = 52$ rules which is much easier to determine. To use such a hierarchical mechanism, a co-ordination scheme is required to combine these behaviours into a single action. Saffiotti (Saffiotti 97) and Tunstel (Tunstel et al. 97] have suggested a fuzzy context rule combination method to perform the high level co-ordination between such behaviours. The context dependent rules are characterised by each behaviour generating *preferences* from the perspective of its goal. Each behaviour has a context of activation, representing the situations where it should be used. The preferences of all behaviours, weighted by a true value of their contexts, are fused to form a collective preference. One command is then chosen from the collective preference.

We use a variant of the method suggested by Saffiotti (Saffiotti 97) and Tunstel (Tunstel et al. 97). In this we will apply fuzzy logic to both implement the individual behaviour elements and the related arbitration (allowing both fixed and dynamic arbitration policies to be implemented) (Hagras et al.

2001a). To achieve this we implement each behaviour as an independent FLC aimed at a simple task, (e.g. edge following or obstacle avoidance) with a resultant small set of inputs and outputs to manage. We chose fuzzy logic to implement the basic behaviours, as it excels in dealing with the kind of imprecise and uncertain knowledge associated with the embedded agent's sensors and actuators.

The outputs of each fuzzy behaviour are fused according to directions supplied by a high-level planner, which may be a person. This fusion itself is a fuzzy process in which different behaviours are co-ordinated to give a coherent output. The high-level planner can also define situations and sequences in which individual behaviours are active, commonly referred to as "willed" operation. The combination of these methods produce a system that is capable of completing complicated tasks from a set of simple tasks, which can be more easily designed than more monolithic alternatives. Fuzzy co-ordination facilitates expression of partial and concurrent activation of behaviours, thereby allowing behaviours to be active concurrently to differing degrees, which gives a smoother control characteristic than switched counterparts (Saffiotti 97). As mentioned earlier, using a hierarchical strategy results in much fewer rules (i.e. much simplified design problem) (Saffiotti 97). In addition, it allows flexible modularised design where new behaviours can be added easily and different tasks achieved by changing the co-ordination parameters (either in willed or automatic mode). Our use of a fuzzy arbitration mechanism, and the flexibility arising, is significant addition to earlier work such as the subsumption architecture (Brooks 92).

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, more information about fuzzy logic can be found in (Lee 90a), Lee 90b).

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

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.

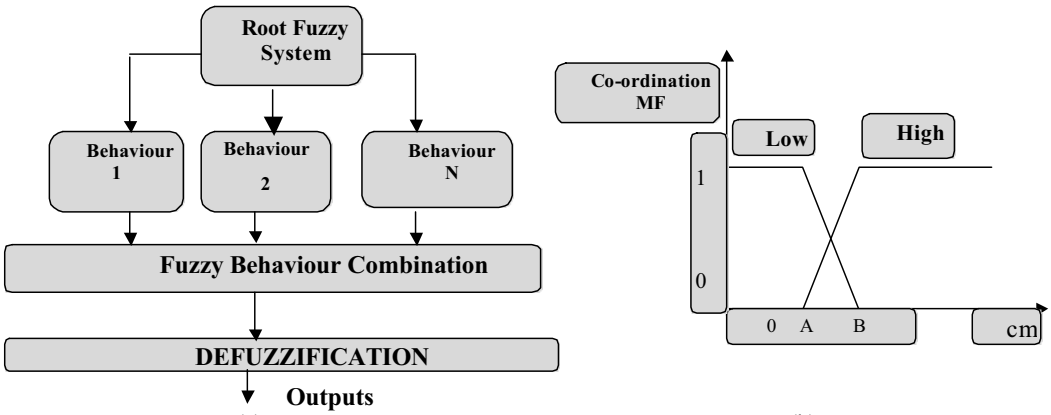



Figure (5): a) Behaviour co-ordination system. b) Membership function for co-ordination parameters.

Ruspini (Ruspini 91) defines fuzzy command fusion as the interpretation of each behaviour-producing unit, acting as an agent, expressing preferences as to which command to apply. Degrees of preferences are represented by a probability distribution (fuzzy in our case) over the command space. In our HFLC architecture a fuzzy operator is used to combine the preferences of different behaviours into a collective preference. Accordingly, command fusion is decomposed into two steps: preference combination and decision-making. In Figure (5-a) each behaviour is treated as an independent fuzzy controller. Using fuzzy behaviour combination, we obtain a collective fuzzy output, which we then defuzzify to obtain a final crisp output. Fuzzy meta-rules or context rules enable *more flexible arbitration* policies to be achieved. These rules have the form IF context THEN behaviour (Saffiotti 97) which means that a behaviour is activated with a strength determined by the context (i.e. a fuzzy-

logic formula). When more than one behaviour is activated, their outputs are fused and each behaviour output scaled by the strength of its context.

In case of using fuzzy numbers for preferences, product-sum combination and height defuzzification, the final output equation, provided by Saffiotti (Saffiotti 97), is given below:

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

Where i represents the behaviours activated by context rules, which can be right/left edge-following behaviour, obstacle-avoidance or goal seeking in case of RA and comfort, safety, economy in case of BA. $Y_t$ is the behaviour command output (robot speed and steering in case of RA and room lighting and heating in case of BA). These vectors are fused in order to produce a single vector $Y_{ht}$ to be applied to the embedded agent. $mm_y$ is the behaviour weight.

In behaviour co-ordination there are a few parameters that must be calculated in the root fuzzy system. In case of robot navigation these are the minimum distance of the front sensors (represented by d1), the minimum distance of the left side sensors (represented by d2), the minimum distance of the right side sensors (represented by d3). The minimum of the fuzzy MF of d1, d2, d3 represented by d4, reflects how obstacle free the robot path is. After calculating these values, each is matched to its membership function as shown in Figure (5-b). These fuzzy values are used as inputs to the context rules which are in case of robot navigation: ***IF d1 IS LOW THEN OBSTACLE AVOIDANCE, IF d2 IS LOW THEN LEFT WALL FOLLOWING, IF d3 IS LOW THEN RIGHT WALL FOLLOWING, IF d4 IS HIGH THEN GOAL SEEKING.***

The context rules determine which behaviour is fired, and to what degree. The final output is calculated using Equation (2). The behaviour weights are calculated dynamically taking into account the situation of the agent. For example, in the robots domain the obstacle avoidance behaviour weight needs to increase as the obstacle comes closer.

## 4. Interactive and Non-Intrusive Learning

In the field of RA, it is preferred that the learning is performed online interactively with the environment. The robot through trial and error evaluates its performance and assigns fitness values to different solutions and it can improve through our patented evolutionary process (Hagras et al. 2001). The robot by discovering its environment can learn by itself the controller needed to achieve the high level objectives and goals specified by the humans and it can update its controller to any environmental and robot kinematics changes it might encounter with no need to repeat the learning cycle. Such online interactive autonomous learning is desired for RAs operating in unstructured, dynamic and changing environments, which is the case of intelligent domestic environments. Such interactive learning allows the robots to program themselves which results in cutting down the costs of reprogramming and making the robots totally autonomous as they need only a high level mission from the humans.

For the BA the situation is different, as the agent needs to autonomously particularise its service to an individual. Building based learning is focused around the actions of people. Buildings are, largely, occupied by people who for a variety of reasons (e.g. time, interest, skills, etc) would not wish, or be able to cope with much interaction with the building systems. Thus in general, learning should as far as possible, be non-intrusive and transparent to the occupants (i.e. requiring minimal involvement from the occupants). The BA are sensor rich and it is difficult to be prescriptive about which sensor parameter set would lead to the most effective learning of any particular action. Thus, to maximise the opportunity for the agent to find an optimum input vector set, whilst containing the processing overloads, the ideal agent would be able to learn to focus on a sub-set of the most relevant inputs.

There are two kinds of online learning and adaptation in intelligent domestic environments one is interactive for RA which is called the Associative Experience Engine presented in Section (5) and the other is non intrusive for the BA which is called Incremental Synchronous Learning (ISL) presented in Section (6).

As the human user is the center of our model the BA agent will use the ISL in a non-intrusive mode to capture the user behaviours. Some of the user behaviours will include the agent identifying any change in the person's behaviour that might signal a need for specific forms of help available via RA. The BA agents will receive high level inputs from the RA such as the robot is near the charger and it will produce high level outputs such as go and fetch a drink based on other input states which can be composite of the building and robot states. The high level output from the BA will be used as a high

level objective function for the RA. The RA will learn and coordinate their basic behaviours such as obstacle avoidance, edge following and goal seeking and they will implement AEE to learn and adapt their controllers to achieve the high level objective.

Figure (6) shows a domestic environment for elderly and patient care which involves cooperation between the BA and the RA which is implemented in the project entitled Care Agents supported by the UK-Korea S&T collaboration fund. In this environment we consider a bed-bound person in a MANUS equipped bed who is served by a Mobile Agent (MA) bringing two commodities to the person's bedside when required. The role of the MANUS robot arm is to serve the patient with the delivered consumables that the MA has brought to a prescribed bedside location. The MA would carry items, such as post, newspapers, tissues, food, drinks, personal items (e.g. hairbrush), tissues and medicines etc. for the use of the patient and deliver them to a fixed point at the bedside so that the goods were readily available to the MANUS arm. The inputs provided to the BA include the current temperature and the lighting and entertainment levels in the room. Also the physical state of the person will be supplied to the BA e.g. prone or sitting up. Another aspect might be certain body signs such as the health monitoring aspects e.g. blood pressure, heart rate etc. Depending on the preferences of the patient and current conditions of the temperature and light, these can be adjusted by the BA. For these preferences to be learnt there would have to be some way of indicating the preferred levels e.g. switches/voice control. The MA provides the BA with information about its power level and about the relative (fuzzy) position of the robot to the bed and to the chargers and the commodities (A and B).

The BA has a rich set of data so that the conditions under which the person elected to ask for the commodity can be captured and recognised. The patient actions will be monitored and his behaviours will be learnt in a non-intrusive manner according to the whole input vector collected from the rest of the sensors. The commands are going to be passed as high level objective functions to the RA, which implement AEE to learn the required controllers to do the job as well as possible. An example of such high level objective is fetch the mail in which the RA will learn and coordinate its basic behaviours such as obstacle avoidance, goal seeking and edge following to go and fetch the mail quickly while avoiding any obstacles. The robot has the capability to adapt its self to any environmental or robot kinematics changes in a short time interval with no need to repeat the learning cycle.

In general we divide the behaviours available to the BA into fixed or dynamic sets, where the dynamic behaviours are learnt from the person, and the fixed behaviours are pre-programmed and include safety and emergency behaviours. These latter behaviours need to be predefined because they cannot easily be learnt. For instance if medicine was going to be administered on a regular basis this would be described explicitly as one of the fixed rules. There will be other safety and emergency behaviours based upon other significant considerations like the temperature at which pipes freeze or what to do in the case of fire and so on. We also think that the BA should never issue pre-emptive commands that would result in the arm moving as such unexpected movement could cause danger to the patient (perhaps the arm needs some safety rules built in to its control system and patient interface). For dynamic behaviours we are going to use a monitoring system, which we call an ISL to record the patient actions, e.g. a request for a drink or turning off the light, and learn to generate rules from this information. These will then be fine-tuned in an incremental and life long mode. This set of behaviours will be able to control, light, temperature, tilt of the bed, sound volume etc and will interface to the MA through high level web based messages with request for specific commodities or advice on recharging itself.

## 5. Associative Experience Engine

In our hierarchical learning procedure we start using a set of working (but not necessarily optimum) fixed membership functions. We then commence learning general rules in each individual behaviour by relating the input sensors to the actuator outputs. In this phase the membership values are not important as the agent learns general rules such as, *if the obstacle is close then turn left.* However in order to achieve a sub-optimal solution for the individual behaviour (a subset of the large search space) we need to find next the most suitable membership functions for the learnt rules. After finding a sub-optimal solution for each behaviour we combine these behaviours and learn the best co-ordination parameters that will give a "good enough" solution for the large search space to satisfy a given mission or plan. Readers are referred to (Hagras et al. 2000a) and (Hagras et al. 2000b) for more information about learning MF and the co-ordination parameters online. After learning the system parameters, the controller then operates in its environment where the online adaptation technique is triggered to adjust the learnt controller to any environmental or kinematics changes or to any new situations it might

encounter. If the controller fails to maintain the desired states, the adaptation technique modifies the poor rules in the relevant behaviours to adjust to the embedded agent differing environmental and kinematics conditions without the need to restart the learning cycle. This is called life long learning where the agent can adapt itself to any new situation and it can update its knowledge about its environment. This hierarchical procedure results in a fast learning time for finding a solution for learning and adaptation in changing unstructured environments.

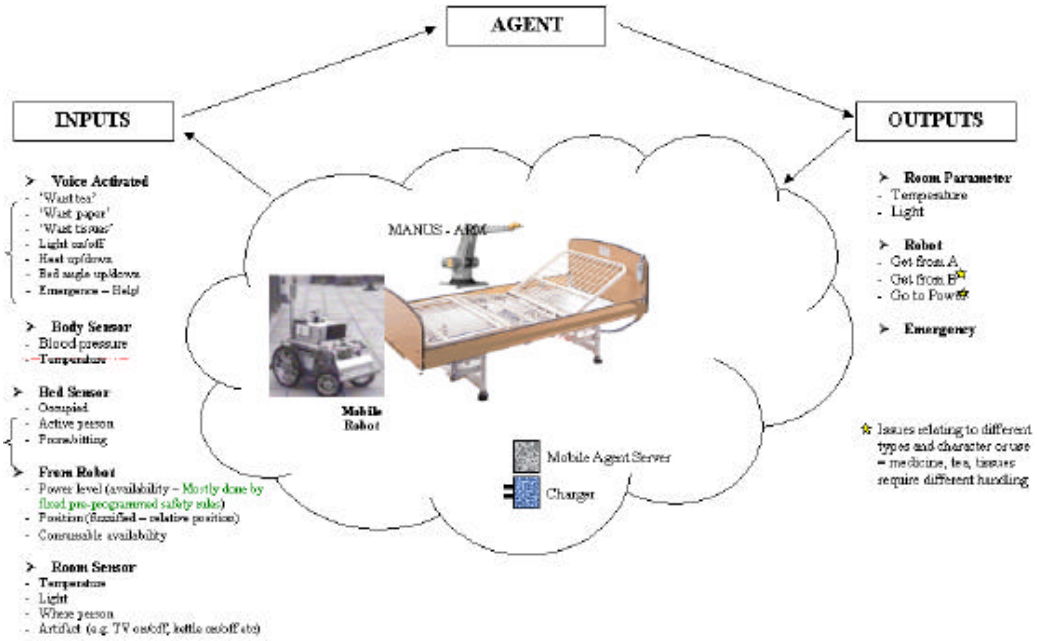



Figure (6): A scenario showing heterogeneous multi embedded agent cooperation in which the robots implement AEE for learning and the building agents implement ISL

Our learning and adaptation techniques are inspired from Nature as in biology, most scientists agree that the remarkable adaptation of some complex organisms comes as a result of the interaction of two processes, working at different time scales: evolution and life long learning. Evolution takes place at the population level and determines the basic structures of the organism. It is a slow process that works by stochastically selecting the better individuals to survive and to reproduce. The life long learning is responsible for some degree of adaptation at the individual level. It works by tuning up the structures, built in accordance with the genetic information, by a process of gradual improvement of the adaptation to the surrounding environment (Rocha et al. 2000). Also condensed learning scenarios over short periods of time differ drastically from continuous learning or life long learning ones as life long learning presents the agent with very different perceptual stimuli than learning in a condensed period of time (Nehmzow 2000). We emulate the natural process by using evolution and online learning to develop a good enough controller of the agent and we use our patented Fuzzy-Genetic system (the *Associative Experience* Engine) described later to speed the slow evolution process. Then we use our online adaptation technique to implement the life long learning where the agent is always updating its knowledge and gaining experience and is able to adapt to its changing environment.

Figure (7) provides an architectural overview of our techniques, which we term as *Associative Experience Engine (AEE).* This forms the learning engine within the control architecture and is the subject of British patent application 99-10539.7. The behaviours are represented as parallel Fuzzy Logic Controllers (FLC) and form the hierarchical fuzzy control architecture presented in the previous section. Each FLC has two modifiable parameters, the *Rule Base* (RB) for each behaviour and the *Membership Functions* (MF). The behaviours receive their inputs from sensors. The output of each FLC is then fed to the actuators via the *Co-ordinator,* which weights their effect. When a behaviour, or collection of behaviours, fail to respond correctly to a situation, a learning cycle is initiated.

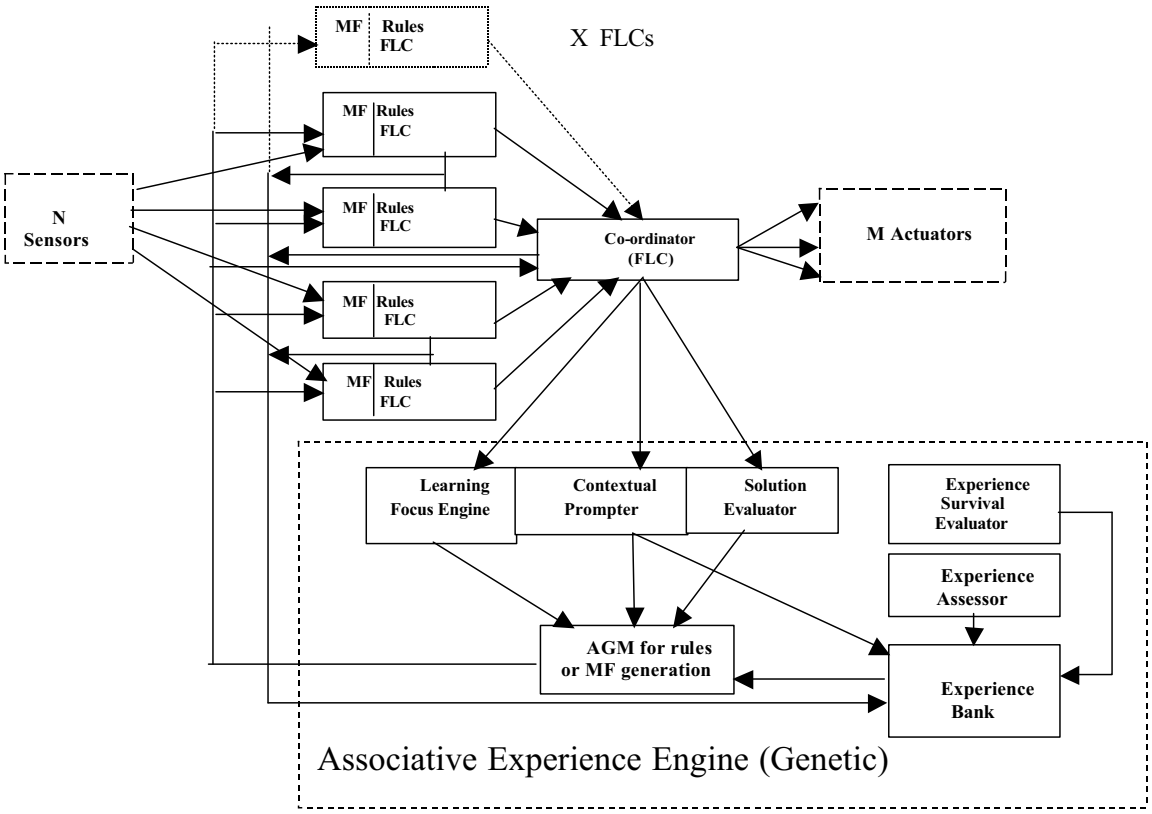



Figure (7): Architectural Overview of Associative Experience Learning Engine (British patent No 99-10539.7)

The learning cycle performed is dependent upon the *Learning Focus,* which is supplied by the co-ordinator according to a higher-level plan. For example, if the *Learning Focus* is to learn the rule base for individual behaviours, then the rule base of each behaviour is learnt alone.

When learning or modifying the rule bases, the learning cycle is sub-divided into local situations. This reduces the size of the model to be learnt. The accent on local models implies the possibility to learn by focusing at each step on small parts of the search space only. The interaction among local models, due to the intersection of neighbouring fuzzy sets causes the local learning to reflect on global performance (Bonarini 99). To further reduce the search space, the system determines if it had encountered similar situations before by checking the stored experiences in the *Experience Bank*. The embedded agent tests different solutions from the experience bank by transferring the "remembered" experiences, which are stored in a queue to the corresponding behaviours. If any of these experiences show success, then they are stored in the FLC and we avoid generating a new solution for our system. An *Experience Assessor* assigns each experience solution a fitness value to indicate the importance of this solution. When the Experience bank becomes full it is the role of the *Experience Survival valuer* to determine which parameters are retained and which are discarded, according to the parameter importance. If the use of past experiences did not solve the situation, we use the highest fitness experience as a starting point for the new learning cycle. We then fire an *Adaptive Genetic Algorithm (AGA*) mechanism using adaptive crossover and mutation parameters, which helps to speed the search for new solutions. The AGA is constrained to produce new solutions in a certain range defined by the *Contextual Constraints* supplied by sensors and defined by the co-ordinator according to the *learning focus*. This avoids the AGA searching places where solutions are not likely to be found. By doing this we narrow the AGA search space to where we are likely to find solutions. The AGA search converges faster as it started from a good point in the search space supplied by the experience recall mechanism and it used adaptive learning parameters and avoided searching regions where solutions are not likely to be found. After generating new solutions wither rules or MF or co-ordination parameters the system tests the new solution and gives it fitness through the *Solution Evaluator* to be discussed next section. The AGA generates new solution until reaching a satisfactory solution.

The online learning mechanism, in addition to the fuzzy behaviours, is also organised as a hierarchy thus leading to one description of this architecture as being a "double-hierarchy". The online learning

mechanisms can be regarded as a hierarchy because there is a tiered set of actions. At the highest level a population of solutions is stored in the *Experience Bank* and tested in a queue. If one of these stored experiences leads to a solution then the search ends, if none of these stored experiences leads to a solution then each of these experiences acquires fitness by the *Experience Assessor* depending how well each solution performed in the situation. The highest fitness experience is used as a starting position to the lower level GA that is used to generate new solutions to the current situation. This hierarchy preserves the system experience, and speeds up the genetic search by starting the genetic algorithm from the best point found in the space.

### 5.1 Learning General Behaviours Rules

The rule base of the behaviour to be learnt is initialised randomly. The designer supplies a preliminary input membership function for each behaviour. As was explained earlier the values of the membership functions are not important as we are seeking general rules. In the following sections we will introduce the various steps of the algorithm to learn the rule base of behaviours that receive immediate reinforcement such as edge following and goal seeking in the robot navigation domain. For learning the rules of behaviours that receive delayed reinforcement please refer to (Hagras 2000c).

After the rule-base initialization, the embedded agent starts operating. If the rule-base contains poor rules then it will begin deviating from its objective. In this case our algorithm is fired to generate new set of rules to correct this deviation. The GA population consists of the most two effective rules in the failure situation. Each chromosome will represent the consequents of one of the effective rules during the bad action. We will use a binary representation. As the case with classifier systems, in order to preserve the system performance, the GA is allowed to replace a subset of the classifiers (the rules in our case). The worst m classifiers are replaced by the m new classifiers created by the application of the GA on the population (Dorigo Colombetti 95]. The new rules are tested by the combined action of the performance and apportionment of credit mechanisms. In our case, only two rule actions will be replaced (those already identified with being predominantly responsible for the deviation).

The system fitness is determined by the *Solution Evaluator* and is evaluated by how much the agent reduces the normalised absolute deviation (d) from the normal value as well as maintaining minimum system oscillation. This is given by:

$$d = \frac{|\ normal.value - deviated.value\ |}{\max .deviation} \qquad (3)$$

Where the normal value will correspond to the value desired by the human designer, which corresponds to the value that gives the maximum normal input membership function. For example, 40 c.m in case of edge following or zero degrees in case of goal seeking (as was specified by the human designer of the membership functions). The deviated value is any value deviating from the normal value. The maximum deviation corresponds to the maximum deviation that can occur as was specified by the human designer membership functions.

The fitness of the solution is given by d1 - d2, where d2 is the normalised absolute deviation before introducing a new solution, and d1 is the normalised absolute deviation following the new solution. The deviation is measured using the agent's sensors, which gives the agent the ability to adapt to the imprecision and noise found in the real sensors rather than relying on estimates from previous simulations.

The fitness of each rule at a given situation is calculated as follows. We can write the crisp output $Y_t$ as in (1). If the agent has N output variables, then we have $Y_{t1}$ ..., $Y_{tn}$ The normalised contribution of each rule p output ($Y_{p1}$, $Y_{p2}$…..$Y_{pn}$) to the total output $Y_{t1}$,$Y_{t2}$…,$Y_{tn}$ can be denoted by $S_{r1}$, . $S_m$ where $S_{r1}$ .. $S_{rn}$ are given by:

$$\mathbf{S_{r1}} = \frac{Y_{p1}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{p=1}^{M}\prod_{i=1}^{G}\alpha_{Aip}} \Big/ Y_{t1} \quad , \quad \mathbf{S_{rn}} \frac{Y_{pn}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{p=1}^{M}\prod_{i=1}^{G}\alpha_{Aip}} \Big/ Y_{tn} \qquad (4)$$

We then calculate each rule's contribution to the final action $S_c = \frac{S_{r1} + S_{r2} + \cdots S_{rn}}{N}$. The most two effective rules are those that have the two greatest values of $S_c$, we use only mutation to generate new solutions because of the small population formed by the fired rules.

### 5.1.1 Memory Application

After determining the rule actions to be replaced according to Equation (4), the robot then matches these rules to sets of rules stored in an *Experience Bank* containing each rule and its best fitness value up to date. The fitness of each rule fired in a given solution is supplied by the *Solution Evaluator* and is given by**:**

$$\mathbf{S_{rt}} = \text{Constant} + (d_1 - d_2) .S_c .(1-F). \quad (5)$$

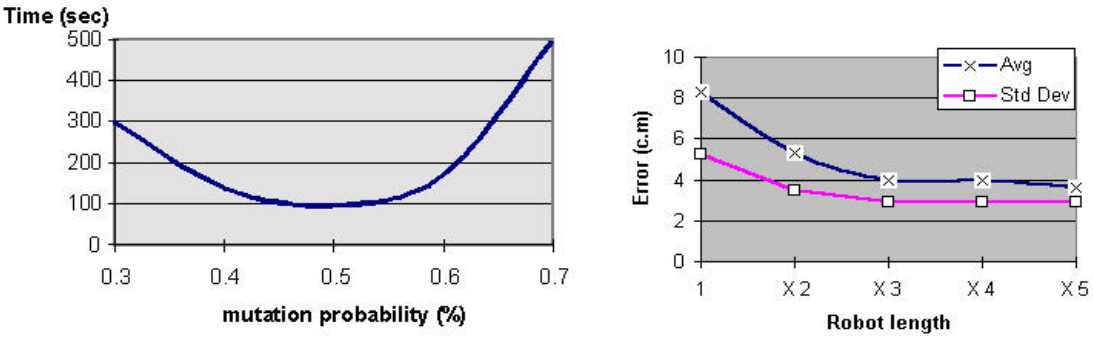
$d_1 - d_2$ is the deviation improvement or degradation caused by the adjusted rule-base produced by the algorithm. If there is improvement in the deviation, then the rules that have contributed most will be given more fitness to boost their actions. If there is degradation then the rules that contributed more must be punished by reducing their fitness w.r.t to the other rules. F is the normalised oscillation in the system actions that can be the system steering and velocity in case of the robots. This makes $S_{rt}$ maximised by maximising the deviation improvement compared to the previous action with minimum adjustments to system output. The variable F was introduced to penalise instant differences between the system outputs to give a steady output and also preserving the actuators by not making large adjustments (ON and OFF) (that might lead to the reduction of the life time of the actuators). The *Experience Survival Valuer* keeps the best fitness solution for each rule in the *Experience Bank*.

After determining the rule actions to be replaced, the agent then matches the effective rules to sets of rules stored in a memory containing each rule and its best fitness value up to date. For each rule action to be replaced, we will replace the current actions in the behaviour rule-base by the best fitness actions stored in the *Experience Bank*. If the deviation decreases, then the agent will keep the best rules in the behaviour rule-base. If the deviation remains the same or increases, the agent uses the GA to produce a new set of solutions. These are obtained by mutating the best fitness chromosomes from the *Experience Bank* to create new actions to replace the most effective rules actions until the deviation begins decreasing or the rule is proved ineffective. This is then considered a solution for the current situation and the rule fitness according to Equation (5) is calculated and is compared with the best fitness rule stored in the memory. If its fitness is greater than the best kept one in the *Experience Bank* then it replaces the best one, otherwise the best one still is kept in the *Experience Bank*. The memory action is supposed to speed up the GA search as it starts the GA from the "best-found" point in the solution space instead of starting from a random point

### 5.1.2 Using GA to Produce New Solutions

The GA begins its search for a new rule action to replace those identified with poor performance by mutating the two best fitness chromosomes from the *Experience Bank* to replace the most two effective rules actions to generate new solutions. The chromosome consists of the effective rule consequents (which can be speed and steering in case of RA). The population consists of two chromosomes corresponding to the two most effective rules. We use a small population, which will give faster convergence to a good enough solution (which is important in online learning) due to the smaller number of individuals to be evaluated and reproduced each generation. However, small populations run the risk of stagnating genetically, i.e. the population doesn't contain sufficient genetic diversity to properly explore the search space. This is why it is important to introduce high mutation rates to introduce new genetic material in the population without running into the risk of end by a random search (Kasabov et al. 99). A mutation rate of 0.5 was chosen after gathering empirical evidence gathered from experimenting with different mutation rates from 0 to 1.0. This was achieved by monitoring the time the embedded agent needed to achieve its goal, as shown in Figure (8-a). It was noticed that at mutation values less than 0.3 there was almost no convergence (which means not finding a good solution) because the population size and the chromosome size are small, and the low mutation rates does not introduce sufficient new genetic material to produce different solutions. The same occurs for high mutation rates (higher than 0.7) as the mutation rate reaches 1.0 the only genetic material available are the primary chromosomes (e.g. 0101) and its inversion (1010), which is not enough for producing any new solutions. Thus 0.5 gave the optimum value for finding a solution. The agent also uses the *Contextual Constraints* supplied by the sensory information to narrow the search space of the GA and thus reduces the learning time. For example, if the robot is implementing left wall following

and it is moving towards the wall, then any action that suggests going to the left will not lead to a solution. Hence, if we use a left side sensor and the robot senses that it is going towards the wall, and then the GA solutions will be constrained not to go left. Hence the solutions generated for the temperature are constrained to be less than the current temperature. This is similar to using a constrained-GA, where the sensor information constrains the GA search space and forces it to look at areas where solutions are likely to be found and avoids doing blind searches in the whole search space, also avoiding abnormal outputs which results from mutation. We have also conducted learning experiments without the *Contextual Constraints*. It was noticed that this increased the learning time compared to those using *Contextual Constraints* by up to four times (average). This is because the GA now has to search in the entire search space, instead of just searching the limited regions where solution are likely to be found.




(a) (b)

Figure (8): a) Mutation rate versus time. b) The relative robot length plotted against the error after learning.

Figure (9) gives an example of a robot deviating from its objective and after trying the memory actions with no success it triggers the GA. In this example the robot has only two output actions (left and right wheel velocities) each one is represented by only 4 fuzzy sets which are *Very Low, Low, Normal, High.* We need only two bits to represent the consequent of each rule thus we have a four bit chromosome. The actions are decoded as follows, *Very Low* is 00, *Low* is 01, *Medium* is 10, *High* is 11. In this example rule 1 and rule 2 were the most effective rules according to Equation (4) and their actions are to be replaced by mutating the actions that obtained the maximum fitness according to Equation (5) which are stored in the *Experience Bank.*

The generation of the new solution is constrained according to the contextual constrains and the mutation continues till the deviation decreases or the effective rules are not effective anymore. After this the fitness of this solution is evaluated using Equation (5) and is compared with the one stored in the *Experience Bank* if it is greater than it, then it replaces the actions stored in the *Experience Bank.* The robot learns this situation and other similar local situations involving different rules. The accent on local models implies the possibility to learn by focusing at each step on small parts of the search space only. The interaction among local models, due to the intersection of neighbouring fuzzy sets causes the local learning to reflect on global performance (Bonarini 99). This reduces the size of the model to be learnt. The agent finishes the learning of the whole behaviour when it reaches a stopping criterion.

### *5.1.3 Stopping Criteria*

The agent assumes it has learnt the rule-base for the behaviour if it succeeds in satisfying the goals specified to its behaviour. In case of the robot navigation, the robot assumes it has learnt the rule-base for the behaviour if it succeeds in maintaining the nominal value for the behaviour for an optimum learning distance. The optimal learning distance has been related to units of length of the robot, so that the algorithm can be applied in an invariant manner to different size robots. In order to determine the optimal learning distance we have conducted numerous experiments using different learning distances corresponding to the robot's length (e.g. 1x robot's length, 2x robot's length, etc.). We then follow the same track that was used during the learning phase to determine the absolute deviation at each control cycle from the optimum value which is maintaining a constant distance from a wall in case of edge following and going towards a goal in case of goal seeking. The average and standard deviation of this error are calculated over 10 experiments over different tracks. From Figure (8-b) it is obvious that the average and standard error for the wall following stabilizes at three times the robot's length at average

value of 4 cm and standard deviation of 3 and become almost the same for a larger number of experiments. Thus we use three times the robot length as our learning length criteria.

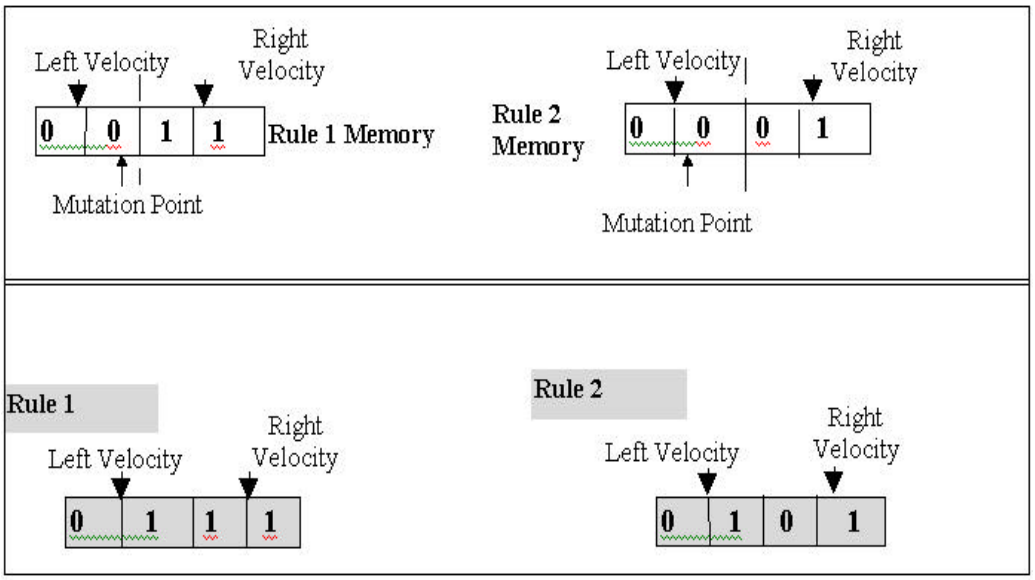



Figure (9): An example of GA processes in which rule 1 and rule 2 consequents stored in Experience Bank and having the best fitness are generating new consequent for rules 1,2 which are the most effective rules responsible for behaviour failure.

### 5.2 Online Adaptation of the learnt controller and Life Long Learning

In the previous sections we explained how to learn the embedded agent controller online and through interaction with the environment using the Evolutionary techniques, imitating the natural evolution and using our patented techniques to speed it up so that we can learn online. In this section we discuss how can we adapt the RA to any environmental changes and implement a life long learning scheme where the agent gains and updates its knowledge by encountering new situations. This means that the agent can add or delete or modify rules according to the situations it encounters.

Also this approach is useful in prototyping as we can learn the embedded agent controller on a prototype agent in a controlled environment and then we transfer it to the agent running in the real environment where we only run a short adaptation cycle with no need to repeat the learning cycle (Hagras et al. 2001a, Hagras et al. 2001b).

Perhaps more importantly, this adaptation and life long learning session is important in the case that environmental or embedded agent characteristics changes occur and the agent needs to rapidly adapt to these changed circumstances.

Although fuzzy logic allows a degree of imprecision to exist within the environment, significant environmental or agent differences will prevent the rule sets from operating correctly. One solution to this problem is to provide a new rule set for each of the different environments, although prior knowledge of the different environments would be needed (which is difficult if not impossible for dynamic unstructured environments). An alternative solution is to allow the existing co-ordinated rule set to be adapted to compensate for the environmental differences. This latter approach only requires a basic rule set to be present from the prototype robot controller and does not require prior knowledge of all possible environments. We start the adaptation from the best-learnt HFLC by the RA, instead of starting from a random point. If the system was started with random rule bases, the system could still modify its actions but would take approximately six times the time needed when started from a good rule base containing some inappropriate rules for the new environment. This figure was found by practical experimentation.

The system discovers the rules (in different behaviours) that, if modified, can lead the embedded agent to adapt to its environment. Whilst it might seem a good idea to change the co-ordination parameters, or the membership functions of the individual behaviours, to adjust the agent behaviour

when it fails, this is not the best approach. This can readily be illustrated by considering a case where the rules in the individual rule bases become inappropriate, then clearly changing the co-ordination parameters will never correct the RA behaviour (as was proved by experimentation). Also changing the MF is a difficult task, as it needs to be done to each individual behaviour necessitating the robot to be taken away from its environment for MF calibration. However, our method allows the poor rules to be found and corrected for each behaviour, online and in-situ without the need to repeat the whole learning cycle, modifying only the actions of a small number of rules that performed poorly.

Imagine a situation where the robot had failed to do its mission. In this case we will modify all the rule bases whose behaviours were active. The agent then uses its short-term memory to return back to its pre-failure position, identifying the two most dominant rules (which can belong to different behaviours). The agent then replaces the actions of these rules to solve this situation. The way the agent determines the dominant rules is done by calculating the contribution of each rule to the final output in a given situation as follows: Apply Equation (2) and substitute $Y_t$ from Equation (1). Then we can write the crisp output $Y_{ht}$ as:

$$Y_{ht} = \frac{\sum_{y=1}^{4} mm_y \dfrac{\sum_{p=1}^{M} y_p \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{\sum_{y=1}^{4} mm_y} \qquad (6)$$

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 of each rule inputs. G is the number of the input variables, $mm_y$ is firing strength of each of the four behaviours.

Because we have N output variables, then we have $Y_{ht1}$, ….$Y_{htn}$. The contribution of each rule p in the behaviour y to the total output $Y_{ht1}$, ..$Y_{htn}$ is denoted by $S_{ra11}$, . $S_{rann}$ where $S_{ra11,}$ $S_{rann}$ are given by:

$$\mathbf{S_{ra11}} = \frac{\dfrac{mm_y}{\sum_{y=1}^{4} mm_y} \cdot \dfrac{Y_{p1} \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{Y_{ht1}}, \quad \mathbf{S_{rann}} = \frac{\dfrac{mm_y}{\sum_{y=1}^{4} mm_y} \cdot \dfrac{Y_{pn} \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}}}{Y_{htn}} \qquad (7)$$

We then calculate each rule's contribution to the final action $Sa_{c1}$ by:

$$Sa_{c1} = \frac{S_{ra11} + S_{ra22} \ldots + S_{rann}}{N}.$$

The most effective rules are those rules with the greatest values of $Sa_{c1}$.

In order to implement the Life-Long learning strategy the agent has a long-term memory composed of all the experiences it encountered in the past and how the agent had succeeded to solve the problems it encountered. These experiences are in the shape of rule clusters having the rule numbers and the consequents the agent had learnt to associate with inputs of these rules in a situation where the agent failed to satisfy the desired behaviour. The rule clusters are arranged in a queue starting from the most recent experiences. These clusters represents how the agent have adapted to different changing environments and situations, i.e. in two different rule clusters we can find the same rule having different consequents which means that each environment should have its different actions. So in case of the agent failing to achieve the desired behaviour in the changing unstructured environment then the actions of the most effective rules can be modified to solve this situation. The agent then matches the effective rules to sets of rules stored in the *Experience Bank*. In the robot domain if, for example; rules 1,2 from the obstacle-avoidance rule base, rule 3 of the left edge-following behaviour and rule 4 of the right edge following need to be replaced, AND the first rule cluster in the Bank contains the consequences of rules 1 of obstacle avoidance; rule 3 of left edge following and rule 6,7 of right edge following, then the consequences of rules 1 for obstacle avoidance and 3 for left edge following will be

changed and rule 2 for obstacle avoidance and 4 for right edge following will remain the same. Then the agent starts operating with this modified rule-base taken from the *Experience Bank*. If it survives and gets out of this situation with no behaviour failures, then these rules are kept in the rule-base of the controller. In this way we have saved the process of learning a solution to this problem from the beginning by using the memorized experience that had worked in different environmental conditions. This is the same as nature where the agent tries to recall its experiences that solved the same situation before. If none of these experiences had solved this situation, the agent assigns to each experience a fitness value indicating how well it performed in the current situation. Then the agent chooses the consequents of the rule clusters that have given the highest fitness and keep them in the rule-bases of the controller in order to serve as a starting position of the Adaptive Genetic Algorithms (AGA) search instead of starting from a random point, these consequents will be the parents of the new rule consequents. This action helps to speed up the genetic search as it starts the search from the best-found point in the search space instead of starting from a random point.

The robot then calculates the fitness value of the solution starting from the modified rule base supplied by the *Experience Bank*. If there is an improvement in the performance produced by the modified rule base, then the rules that contributed most must be given increased fitness to boost their actions. If there is no improvement then the rules that contributed most must be punished by reducing their fitness w.r.t to other rules and the solutions that had small contributing actions are examined. The fitness of each rule is supplied by the *Solution Evaluator* and is given by:

$$\mathbf{S_{rat}}_1 = \text{Constant} + (d_{new}-d_{old}) .S_{ac1}. (1-F) \qquad (8)$$

Where $d_{new}$ is the new performance of the robot (using the modified rule base, $d_{old}$ is the old performance of the agent before modifying the rule base, $d_{new}-d_{old}$ is the performance improvement or degradation caused by the adjusted rule-base produced by the algorithm. F is the normalised oscillation in the system actions. Thus $S_{rat1}$ is maximised by improving the robot performance before behaviour failure with minimum oscillations in the system output. In the first population of the GA, because there is no performance assessed yet, we blame the rules that have contributed most to the behaviour failure. The fitness of each rule is given by:

$$\mathbf{S_{rat}} = \text{Constant} - S_{ac1} \qquad (9)$$

The performance of the robot is assessed according to the failure of the robot to reach or maintain a certain target. For example in case the robot approaches an obstacle to a dangerous distance then the performance of the robot will be assessed by how it increases the distance moved before collision.

However, a problem occurs as the system begins accumulating experience that exceeds the physical memory limits. This implies that we must delete some of the stored information. To deal with this, for every rule cluster we attach a *difficulty counter* to count the number of iterations taken by the agent to find a solution to a given situation, we also attach a *frequency counter* to count how frequently this rule have been retrieved. The *degree of importance* of each rule cluster is calculated by the *Experience Survival Valuer* based on the product of the *frequency counter* and the *difficulty counter*. This attempts to keep the rules that required a lot of effort to learn (due to the difficulty of the situation) and also the rules that are used frequently. When there is no more room in the *Experience Bank*, the rule cluster that had the least *degree of importance* is selected for replacement. If two rule clusters share the same importance degree, tie-breaking is resolved by a least-recently-used strategy. The rule that has not been used for the longest period of time is replaced. Thus an *age parameter* is also needed for each rule cluster. The value of the age parameter increases over time, but is initialised whenever the associated cluster is accessed. The limit for the memory clusters is set to 2000 rule clusters, so if we exceed this limit the *Experience Survival Valuer* begins using the *degree of importance* and the age operator to optimise the memory.

The AGA is using the Srinivas method (Srinivas and Patnaik 96). If, for example in the robot domain, rule number 5 of the obstacle avoidance and rule 7 of the left wall following are chosen for reproduction by roulette wheel selection due their high fitness. This implies they have either contributed more with their actions to the final action that caused improvement, or contributed less with their actions to final action that caused degradation. Then we apply adaptive crossover and mutation operators to both chromosomes. The resultant offspring will be used to replace the consequent of rules 1 and 2 for obstacle avoidance that were largely blamed for the behaviour failure. The AGA is constrained to produce offspring according to the *Contextual Constraints* supplied by the input sensors. The robot ends the adaptation when the agent achieves the desired response.

## 6. Incremental Synchronous Learning for non-intrusive learning

The Incremental Synchronous Learning (ISL) architecture is shown in Figure (10). The BA need to address somewhat different learning needs, short "initialisation" and long "life-long" learning. In general a learning mechanism within a building should be non-intrusive.

The ISL forms the learning engine for the BA. 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, in the case of an Intelligent Building (IB) these could be - the Safety, Emergency and Economy behaviours or dynamic and adaptable such as, in an IB, Comfort behaviours (i.e. behaviours adapted according to the occupant's actual behaviour).

Each dynamic FLC has one parameter that can be modified which is the *Rule Base* (RB) for each behaviour.

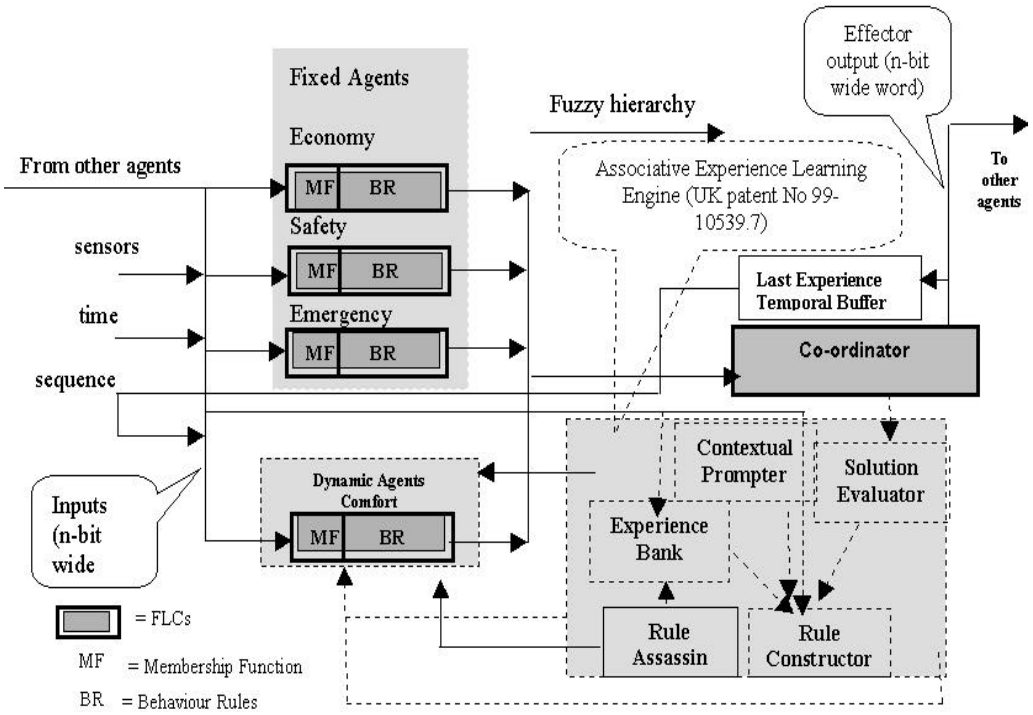



Figure (10): The ISL Embedded-Agent Architecture

Also, at the high level the co-ordination parameters can be learnt (Hagras et al. 2000b). The HFLC is the same as explained in Section (3). 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 a 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 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 lifetime of the use of environment, thus forming a life-long learning phase.

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 99). 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 normalised contribution of each rule p output ($Y_{pN}$) to the total output $Y_{ht}$ can be denoted by $S_{ra11}$ and is given by Equation (7).

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 occupant's 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 99). So we can have global results coming from the combination of local models, and smooth transition between close models. It is necessary to point to a significant difference in our method of classifying or managing rules, which is, rather than seeking to extract generalised rules we are trying to define particularised rules.

After the initial Monitoring initialisation 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 the briefly to 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-based 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 long times to learn as much needed rules which is highly needed in care houses. Also the system can start with short Monitoring time and then jump to the life long learning mode to learn the user behaviour quickly, which is needed in hotel rooms. The weighting is set by adjusting the Monitoring non-intrusive period to the required application.

The emphasis on particularisation over generalisation can create problems when the particularised rules storage needs of different users exceed the physical memory limits. This implies that we must remove some stored information. To manage this we employ various mechanisms. One such mechanism is for every individual rule base cluster we have, we attach a *difficulty counter* to count the time taken by the agent to learn a particular rule base. We also attach a *frequency counter* to count how often this rule base has been retrieved. The *degree of importance* of each rule base cluster is calculated by the *Rule assassin* and is given by the product of the *frequency counter* and the *difficulty counter*. This approach tries to keep the rules that have required a lot of effort to learn (due to the difficulty of the situation) and also the rules that are used frequently. When there is no more room in the *Experience*

*Bank*, the rule base cluster that had the least *degree of importance* is selected for removal by the *Rule assassin*. If two rule base clusters share the same importance degree, tie-breaking is resolved by a least-recently-used strategy; derived from a "life invocation" flag that is updated each time a rule is activated.

Multi-Agent operation 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. 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.

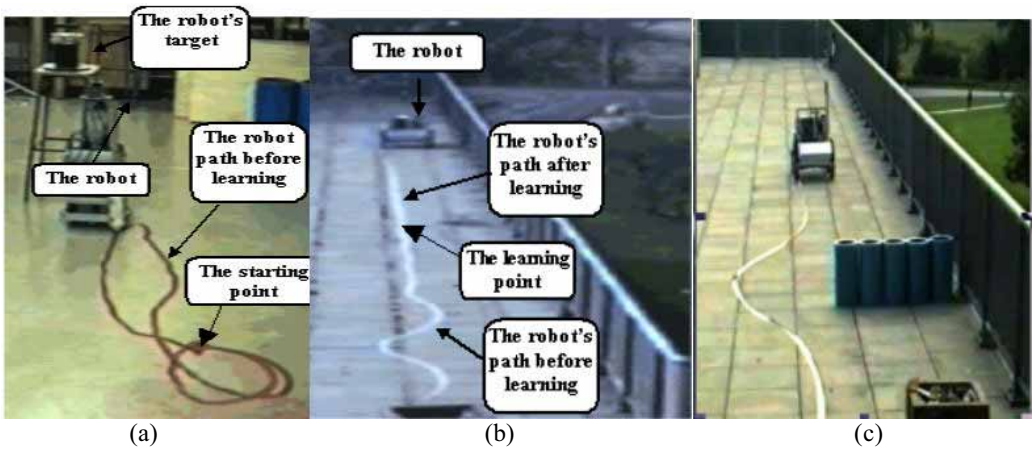
## 7. EXPERIMENTS AND RESULTS

This section describes the experiments and results of applying the AEE for online learning and adaptation in mobile RA and the application of ISL to BA and how the Building and Robotic agents communicate and cooperate to do the specified jobs.

All the robots will have the same sensors, there will be three front sonar sensors for obstacle avoidance and two sonar sensors on each side of the robot for edge following, each sensor is represented by three membership (which are Near, Medium, Far) as this was found to be the least number of membership function to give a satisfactory result. Also the robots have sensors to measure the bearing from the goal. Each bearing sensor is represented by seven memberships as this was found to be the least number of membership functions to give a satisfactory result. The output membership functions will be represented by four membership functions (which are Very Low, Low, Medium and High).

The following experiments shows the RA using the AEE to learn their basic behaviours which are goal seeking, edge following and obstacle avoidance. We had used different sizes of robots to show that our techniques are robot independent and also we performed some experiments in outdoor environments to test that the AEE can deal with highly difficult and unstructured environments.

Figure (11-a) shows an indoor robot learning the rule base of the goal seeking behaviour, using an imprecise infrared beacon system to emulate a goal. The robot learnt the rule-base in an average of 96 seconds (this time is not the actual time to find a solution but it is bounded by the robot microprocessor speed and the robot speed) over 5 trials using a 68020 microprocessor robot with maximum speed of 13cm/sec starting from different positions with different initialised rule bases (the average error over more experiments was found to converge after 5 trials).




(a) (b) (c)

Figure (11): a) Indoor robot learning goal seeking behaviour. b) The outdoor robot learning the edge following behaviour. c) The outdoor robot learning the obstacle avoidance behaviour.

Figure (11-b) shows the electrical outdoor robot learning the rule-base for the right edge following behaviour of an irregular metallic fence, the robot tried to follow the edge keeping a desired distance of

1 m. The figure shows the robot reducing its deviation rapidly until it succeeds in maintaining the robot with a very small deviation. The robot's path after 79 seconds of learning is very smooth despite the irregularity of the fence and the highly imprecise sensors. Table (1) shows the learnt rule base that adapts the robot to the edge irregularity while taking into account the robots kinematics and dynamics. The robot followed the edge in almost a straight line while avoiding any bigger bumps in the fence. The rule-base was initialised randomly with the first 4 rules actions assigned to move forward. The actions of the fifth and the sixth and the seventh rules were assigned to turn left. And the actions of the remaining rules were assigned to turn right. Some of the rules look contradictory to human reasoning but they fit the robot. For example in the rule stating *if* Right Front Sensor (*RFS) is Near and Right Back Sensor (RBS) is Near,* a human consequent would be to turn left to avoid collision with the fence but the learning system had chosen to turn right towards the fence. The reason for this is that the robots dynamics are biased to the left and when the robot is activating rules 2 and 3, the robot turns sharply to the left and the robot takes time in order to stabilise again, thus causing a zigzag path. The first rule consequent had resulted in damping the combined actions composed of these rules and thus resulting in a smooth path with minimum deviation rather than a zigzag path. Also the seventh rule states if *RFS is Far and RBS Near, which* is a corner situation, a human decision would be to turn right to follow the fence and turn around this corner. The learnt rule suggests going to the left away from the fence to give the robot more space until the robot goes to the next rule which is *RFS is Far and RBS Med* then the robot turns around the corner smoothly as it has enough space to perform the turning. The learnt rule-base had produced an average deviation from the desired distance of 4.3 cm with a standard deviation of 1.5 while the human designed rule-base had resulted in average deviation of 27 cm and a standard deviation of 8.

| Rule No | RFS | RBS | Speed | Steering |
|---|---|---|---|---|
| 1 | NEAR | NEAR | LOW | HIGH |
| 2 | NEAR | MED | MED | LOW |
| 3 | NEAR | FAR | MED | LOW |
| 4 | MED | NEAR | LOW | VERY LOW |
| 5 | MED | MED | MED | MED |
| 6 | MED | FAR | MED | MED |
| 7 | FAR | NEAR | LOW | VERY LOW |
| 8 | FAR | MED | LOW | HIGH |
| 9 | FAR | FAR | MED | HIGH |

Table (1): The best learnt rule base of the right edge following behaviour using outdoor robots.

Figure (11-c) shows the outdoor robot learning the obstacle avoidance behaviour while following an irregular metallic fence full of bumps in an outdoor environment at a desired distance of 120 cm whilst avoiding obstacles at a distance of 1m. The robot converged to a solution after an average of 6 iterations taking 3 minutes of robot time. It gave a small average deviation of 2.4 cm and standard deviation of 0.52 for the desired safe distance. After learning the rule bases the robot then learns the best membership functions that suits the learnt rules to form a sub optimal behaviour. The robot then learns the best coordination parameters to satisfy the high level mission to form the HFLC. More information about learning the MF and the coordination parameters online can be found in (Hagras et al. 2000a and Hagras et al. 2000b]

What were learnt are HFLC, which were sub-optimal under certain environmental and robot kinematics conditions. If the environmental or the robot kinematics changed dramatically these parameters would not be sub-optimal and the learning cycle would have to be repeated from the beginning. Of course this is not practical in highly and unstructured dynamic environments such as the domestic environments where the environment is continuously changing. Although fuzzy logic allows a degree of imprecision to exist within the environment, significant environmental differences will prevent the rule sets from operating correctly. One solution to this problem is to provide a new rule set for each of the different environments, although prior knowledge of the different environments would be needed (which is difficult if not impossible for dynamic outdoor environments). An alternative solution is to allow the existing coordinated rule set to be adapted to compensate for the environmental differences. This latter approach only requires a basic rule set to be present and does not require prior knowledge of all possible environments. The online adaptation technique will discover the rules in

different behaviours that, if modified online and in-situ, without the need to repeat the whole learning cycle can lead the robot to adapt to its environment [Hagras et al. 2001b].

In figure (12-a) the robot was given a high level mission which is to align to center of the corridor while avoiding any obstacles and going to its goal after exiting the corridor, this high level mission can be supplied by the BA. In this figure we had destroyed the rules in all behaviours by setting all the rule consequences to "go right", thus causing the robot to collide with walls in the corridor, we also tried loosening the left wheel to simulate changing robot conditions. Also the rules in the goal seeking behaviour were destroyed to make the robot always shift to the right. These changes were done to simulate changes in the environmental and robot kinematics conditions and to test if the life long learning strategy within the AEE can adapt the robot online with no need to repeat the learning phase. It took the robot 8 minutes (including reversing time) to get out of the corridor and to modify the rule bases of the co-ordinated behaviours. It modified 7 rules in the obstacle avoidance behaviour, 4 rules in the left wall following behaviour, 3 rules in the right wall following behaviour and 3 rules in the goal seeking behaviour (i.e. 17 rules). It learnt these rules in an average of 20 iterations (episodes) over 4 experiments. The modified rules are shown in Table (2).

Figure (12-c) shows the robot after learning the sub optimal HFLC co-ordinating the obstacle avoidance behaviour and the goal seeking behaviour. The robot was required to go up hill to go to target 1 and then go down hill to target 2 and then it had to repeat the path back up hill to target 1, the operation was repeated continuously for 2 hours. This operation is similar to the missions described before in which the BA gives a high level mission of go to target 1 and target 2 while avoiding obstacles. While going uphill the online adaptation module had to change the goal seeking behaviour speed consequents to High to give the robot more power to go up this steep hill, as the robot was trained on a nearly flat ground. When going down hill the goal seeking consequents had to be changed again to Low. Although the robots are equipped with wheel encoders and a speed controller however there is a need for a high level set point from the goal seeking behaviour to give more power to the robot up hill and to protect it from slipping when going downhill. When the robots goes uphill again it doesn't need to relearn the rule base, thanks to the Experience Bank which allows the robots to recall its past experience rather than relearning. The robot path after two hours is nearly the same with an average deviation from the original path of 2.5 cm and standard deviation of 1.2.

| | **LFS** | **MFS** | **RFS** | **Left Speed** | **Right Speed** |
|---|---|---|---|---|---|
| **Obstacle Avoidance** | Near | Far | Near | Med | Med |
| | Med | Near | Near | Very Low | High |
| | Med | Med | Near | Low | High |
| | Med | High | Near | Low | Med |
| | Far | Far | Near | Very Low | High |
| | High | Med | Near | Low | High |
| | High | High | Near | Very Low | Med |
| | **LSF** | **LSB** | **Left Speed** | **Right Speed** | |
| **Left Edge Following** | Med | Near | Low | Med | |
| | Far | Near | Very Low | High | |
| | High | Med | Med | High | |
| | High | High | Med | High | |
| | **RSF** | **RSB** | **Left Speed** | **Right Speed** | |
| **Right Edge Following** | Near | Near | Very Low | High | |
| | Low | Med | Low | High | |
| | Low | High | Very Low | High | |
| **Goal Seeking** | **Bearing** | | **Left Speed** | **Right Speed** | |
| | Very Very Negative | | Very Low | High | |
| | Very Negative | | Low | High | |
| | Med Negative | | Med | High | |

Table (2): The modified rules for the experiment in Figure (12-a).

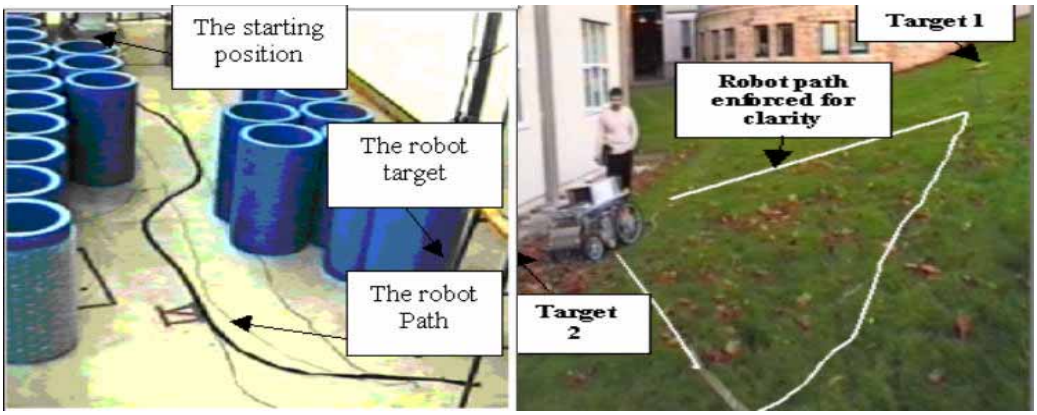
We have conducted a number of experiments using the ISL to learn and adapt the behaviours of different users with different tastes and desires. For each experiment the users have spent up to 5 hours in the iDorm.

For the experiments the room had four environmental parameters to control:

- lighting level (I1) represented by 3 triangular fuzzy sets (Low Norm High)
- temperature (I2) represented by 3 triangular fuzzy sets (Low Norm High)
- outside level of lighting (I3) represented by three triangular fuzzy sets (Bright Dim Dark)
- user location represented by two fuzzy sets (I4) wither he is on the Desk or lying on the Bed.
- There were seven outputs to control;
- two dimmable spot lights above the desk (O1) represented by five triangular fuzzy sets (VLow, Low, Norm, High, VHigh).
- two dimmable lights above the Bed (O2) again represented by five triangular fuzzy sets
- a Bedside Lamp (O3) represented by ON-OFF Fuzzy states;
- a Desk Lamp (O4) represented by ON-OFF Fuzzy states
- automatic blinds whose opening can be controlled (O5) 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 Norm is 50 % opening
- a Fan Heater (O6) represented by ON-OFF Fuzzy states and a Fan Cooler (O7) represented by ON-OFF Fuzzy states.

This forms a rule base of 3*3*3*2 =54 rules. However as the ISL learns only the rules required by the user we find that we don't need to learn the whole rule base, thus the rule base will be optimised.

Table (1) shows the learnt rule-base for “User-1” pictured in Figure (13) who occupied the room for more than two hours. In these experiments the users 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 being sometimes used. He would also close the blinds in the evening using combinations of the ceiling lamps and desk lamp. The data also contains “going to sleep behaviours” including reading before sleeping and getting up spontaneously at night to work (they are students!).




(a) (b)

Figure (12): The indoor robot using life long learning to adapt itself. b) The outdoor robot

implementing continuous learning.

Figure (13): “User 1 in the iDorm”.

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 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 Norm and switch ON the desk and the bedside lamp and the blind to Norm 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 (3) 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 the input parameters required by the user and hence giving more optimisation to the system.

| I1 | I2 | I3 | I4 | O1 | O2 | O3 | O4 | O5 | O6 | O7 |
|---|---|---|---|---|---|---|---|---|---|---|
| XX | Norm | Bright | Desk | VHigh | VHigh | OFF | OFF | Norm | OFF | OFF |
| XX | High | Bright | Desk | VHigh | VHigh | OFF | OFF | Norm | OFF | OFF |
| High | Low | Dim | Desk | VHigh | VHigh | OFF | OFF | Norm | OFF | OFF |
| Norm | Norm | Bright | Bed | VHigh | VLow | ON | OFF | VLow | OFF | OFF |
| Norm | High | Bright | Bed | VHigh | VLow | ON | OFF | VLow | OFF | OFF |
| High | Norm | Bright | Bed | VHigh | VLow | ON | OFF | VLow | OFF | OFF |
| High | High | Bright | Bed | VHigh | VLow | ON | OFF | VLow | OFF | OFF |
| Norm | Norm | Dark | Desk | Norm | Norm | ON | ON | Norm | OFF | OFF |
| Norm | High | Dark | Desk | Norm | Norm | ON | ON | Norm | OFF | OFF |
| High | Norm | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | ON | OFF |
| High | High | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | ON | OFF |
| Low | Norm | Dark | Bed | VLow | VLow | OFF | OFF | VLow | OFF | OFF |
| Low | High | Dark | Bed | VLow | VLow | OFF | OFF | VLow | OFF | OFF |
| Low | Norm | Dark | Desk | Norm | Norm | ON | ON | Norm | OFF | OFF |
| Low | High | Dark | Desk | Norm | Norm | ON | ON | Norm | OFF | OFF |

Table (3): The learnt Rule Base for “User-1”

Table (4) shows the learnt rule base for another user with different desires and taste. His Experiments were conducted at night giving rise to 12 learnt rules. 8 of which were generated during the Initialisation phase, 2 rules being ported from a similar user and were satisfactory to the user. The last 2 rules were tuned during the life long learning period as when sitting on a Desk and reading he wanted the desk light to be ON while for the similar user it was OFF so again it is a small fine tuning to satisfy the user which verifies the use of the Experience Bank.

Of course the learnt rules in both tables are different as our agent particularises rather than generalises. To adequately show all the characteristics of this agent we would need to run experiments over much longer periods (e.g. 12 months to get a full environment variation).

| I1 | I2 | I3 | I4 | O1 | O2 | O3 | O4 | O5 | O6 | O7 |
|---|---|---|---|---|---|---|---|---|---|---|
| Norm | Norm | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| Norm | High | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| High | Norm | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| High | High | Dark | Desk | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| Norm | Norm | Dark | Bed | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| Norm | High | Dark | Bed | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| High | Norm | Dark | Bed | VHigh | VHigh | OFF | ON | VHigh | OFF | OFF |
| High | High | Dark | Bed | VHigh | VHigh | ON | OFF | VHigh | OFF | OFF |
| Low | Norm | Dark | Bed | VHigh | VHigh | OFF | ON | VHigh | OFF | OFF |
| Low | High | Dark | Bed | VHigh | VHigh | OFF | ON | VHigh | OFF | OFF |
| Low | Norm | Dark | Desk | VHigh | VHigh | OFF | ON | VHigh | OFF | OFF |
| Low | High | Dark | Desk | VHigh | VHigh | OFF | ON | VHigh | OFF | OFF |

Table (4): The learnt Rule Base for "User-2"

We have compared this method, which is an online proactive method, with other offline supervised learning systems such as the Mendel-Wang ANFIS approach (Mendel and Wang 92). We found, from the experimental results, that our system gives comparable results to that of such offline approaches. However our system has the added advantage of being on-line, adaptable to new users and able to particularise rather than generalise. Offline methods have to repeat the learning cycle from the beginning and require that the initial training set, plus any newly acquired data, be used. Our system works by cause-effect actions in the form of fuzzy rules, based on the occupant's actions. The advantage of this is that the system responds and adapts to the users needs in an immediate manner. More information about comparison with other techniques are listed in (Hagras et al. 2000d, Hagras et al. 2002].

Figure (14) shows the BA communicating with the RA agents in the form of a Manus arm to fetch a newspaper from the table and deliver it to the user in the bed. These experiments were conducted in cooperation with KAIST and in the Human-friendly Welfare Robotic system lab in Daejeon-Korea. The RA only receives a high level mission of getting the newspaper which is a high level output from the ISL.

Figure (14): The Building agent communicates with the Robotic agents to get the newspaper to the bed.

## 8. Conclusions

In this chapter we have presented our learning and adaptation techniques applied in intelligent domestic environments. We the system operating in both an interactive mode for Robotic agents and in a non-intrusive mode for Building agents. We have also illustrated how to form a heterogeneous multi

embedded agent system in which the various member agents cooperate and communicate to satisfy the user desires and objectives. The intelligent and autonomous system we have described learns behaviours and adapts online and through interaction with the environment, including its occupants.. 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).

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 climate cycle), significantly expand the sensor-effector set and explore more fine-grained and course grained distributed embedded-agents (e.g. with agents within gadgets, or communicating rooms). We are also currently conducting experiments which link agents in physically separated intelligent spaces together (eg UK to Korea) which are allowing us to explore us to explore the issues in virtual adjacency across time and culture zones.

## Acknowledgement

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

We are pleased to acknowledge the contribution of Malcolm Lear, Robin Dowling and Arran Holmes for their help building the intelligent dormitory and intelligent-artifacts. We would also like to thank, Anthony Pounds-Cornish, Sue Sharples, Gillian Kearney, Filiz Cayci, Adam King, and Fiayaz Doctor for their indirect contributions arising from many stimulating discussions on intelligent-artifact and embedded-agent issues.

## References

[Angelov 2000] P. Angelov, R. Buswell, V. Hanby, " Automatic Generation of Fuzzy Rule-based Models from Data by Genetic

[Bonarini 99] A. Bonarini, "Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier systems", Proceedings of the Genetic and Evolutionary Computation Conference, pp. 52-60, Orlando, Florida, July, 1999.

[Brooks 92] R. Brooks, "Artificial Life and Real Robots", MIT press 1992.

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

[Callaghan et al 2000] V.Callaghan, G.Clarke, M.Colley, H.Hagras, "A Soft-Computing based DAI Architecture for Intelligent Buildings" to appear in the book series entitled Studies in Fuzziness and Soft Computing, Springer Verlag, August 2000.

[Callaghan et al 2001] Victor Callaghan, Graham Clarke, Martin Colley and Hani Hagras, " Embedding Intelligence: Research Issues for Ubiquitous Computing" Proceedings of the Ubiquitous Computing in Domestic Environments Conference, 13-14 September 2001, Nottingham-UK.

[Colley 2001] M.Colley, G. Clarke, H.Hagras, V. Callaghan, "Integrated Intelligent Environments: Cooperative Robotics & Buildings" Proceedings of the 32nd International Symposium on Robotics and Automation, pp. 745-750, Seoul-Korea, April 2001.

[Davidsson 98] P. Davidsson "Energy Saving and Value Added Services - Controlling Intelligent-Buildings Using a Multi-Agent System Approach" in DA/DSM Europe DistribuTECH, Penn Well, 1998.

[Delgado 2000] M. Delgado, F. Zuben, F. Gomide, " Evolutionary design of Takagi-Sugeno Fuzzy Systems: a Modular and Hierarchical Approach", Proceedings of the 2000 IEEE International Conference on Fuzzy Systems, San Antonio-USA, May 2000.

[Dorigo 95] M. Dorigo, M. Colombetti, "Robot Shaping: Developing Autonomous agents through learning", Artificial Intelligence Journal, Vol.71, pp. 321-370, 1995.

[Hagras 2001b] H.Hagras, M. Colley, V. Callaghan, "Life Long Learning and Adaptation for Embedded agents operating in unstructured Environments", Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference Vancouver, Canada, July 2001

[Hagras 2000a] H. Hagras, V. Callaghan, M. Colley, "On Line Calibration of the sensors Fuzzy Membership Functions in Autonomous Mobile Robots" to appear in the Proceedings of the 2000 IEEE International Conference on Robotics and Automation, pp. 3233-3238, San Francisco-USA, April 2000.

[Hagras 2000b] H. Hagras, V. Callaghan, M. Colley, "Learning Fuzzy Behaviour Co-ordination for Autonomous Multi-Agents Online using Genetic Algorithms & Real-Time Interaction with the Environment " to be appear in the Proceedings of the 2000 IEEE International Conference on Fuzzy Systems, pp. 853-859, San Antonio-USA, May 2000.

[Hagras 2000c] H. Hagras, "An Evolutionary Computing Based Technique for Embedded-Agent System Design for Mobile Robots", PhD thesis, University of Essex, March 2000.

[Hagras 2000d] H.Hagras, V.Callaghan, M.Colley, G.Clarke, "A Hierarchical Fuzzy Genetic Multi-Agent Architecture for Intelligent Buildings Sensing and Control " Proceedings of the International Conference on Recent Advances in Soft Computing 2000, pp. 199-206, Leicester-UK, June 2000.

[Hagras et al 2001] H. Hagras, V. Callaghan, M. Colley, "Prototyping Design and Learning in Outdoor Mobile Robots operating in unstructured outdoor environments" IEEE International Robotics and Automation Magazine, Vol. 8, No.3, pp.53-69, September 2001.

[Hagras et al 2002] H. Hagras, V. Callaghan, M.Colley, G. Clarke, "A Hierarchical Fuzzy Genetic Multi-Agent Architecture for Intelligent Buildings Learning, Adaptation and Control", to appear in the International Journal of Information Sciences, August 2002.

[Hoffmann 98] F. Hoffmann, "Incremental Tuning of Fuzzy Controllers by means of Evolution Strategy", Proceedings of the GP-98 Conference, pp. 550-556, Madison, Wisconsin, 1998.

[Kasabov 99] N. Kasabov, M. Watts, "Neuro-Genetic Information Processing for Optimisation and Adaptation in Intelligent Systems", In: Neuro-Fuzzy Techniques for Intelligent Information Processing, N.Kasabov and R. Kozma, Eds. Heidelberg Physica Verlag, pp. 97-110, 1999.

[Kasabov 99] N. Kasabov, M. Watts, "Neuro-Genetic Information Processing for Optimisation and Adaptation in Intelligent Systems", In: Neuro-Fuzzy Techniques for Intelligent Information Processing, N.Kasabov and R. Kozma, Eds. Heidelberg Physica Verlag, pp. 97-110, 1999.

[Linkens 95] G. Linkens, O. Nyongeso, "Genetic Algorithms for Fuzzy Control, Part II: Online System Development and Application", IEE proceedings Control theory applications, Vol.142, pp.177-185, 1995.

[Matellan 98] V. Matellan, C. Fernandez, J. Molina, "Genetic Learning for Fuzzy Reactive Controllers", Journal of Robotics and Autonomous Systems", Vol. 25, pp. 33-41, 1998.

[Mendel 92] J. Mendel, L.Wang, "Generating Fuzzy Rules by Learning Through Examples, IEEE Transactions on Systems, Man and Cybernetics", vol .2 pp. 1414-1427, 1992.

[Miglino 95] O. Miglino, H. Lund, S. Nolfi, "Evolving Mobile Robots in Simulated and Real Environments", Technical report NSAL-95007, Reparto di sistemi Neurali e vita Artificale, Instituto di Psicologia, Consiglio Nazionale delle Ricerche, Roma, 1995.

[Minar 99] N. Minar, M. Gray, O. Roup, R. Krikorian and P Maes. "HIVE: Distributed Agents for Networking Things". MIT Media Lab, Appeared in ASA/MA, 1999.

[Mozer 98] 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.

[Nehmzow 2000] U. Nehmzow, " Continuos Operation and perpetual Learning in Mobile Robots", Proceedings of the International Workshop on Recent Advances in Mobile Robots, Leicester, UK, June 2000.

[Pedrycz 98] W. Pedrycz, F. Gomide, " An Introduction to Fuzzy Sets: Analysis and Design", MIT press, Cambridge 1998.

[Rocha 2000] M. Rocha, P. Cortez, J. Neves, " The Relationship between Learning and Evolution in Static and Dynamic Environments", Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, Paisley-UK, pp. 500-506, June 2000.

[Ruspini 91] E. Ruspini, "Truth as Utility a Conceptual Synthesis", In Proceedings of the 7th Conference on uncertainty in Artificial Intelligence, pp. 458-464, Los Angeles, CA 1991.

[Saffiotti 97] A. Saffiotti, "Fuzzy Logic in Autonomous Robotics: Behaviour Co-ordination", Proceedings of the 6th IEEE International Conference on Fuzzy Systems, Vol.1, pp. 573-578, Barcelona, Spain, 1997.

[Sherwin 99] A Sherwin. Internet Home Offers a Life of Virtual Luxury. The Times, pp. 10, 3rd Nov 1999.

[Sousa 2000] M. Sousa, M. Madrid, " Optimisation of Takagi-Sugeno Fuzzy Controllers Using a Genetic Algorithm", Proceedings of the 2000 IEEE International Conference on Fuzzy Systems, San Antonio-USA, May 2000.

[Srinivas 96] M. Srinivas, L. Patnaik, "Adaptation in Genetic Algorithms", Genetic Algorithms for pattern recognition, Eds, S.Pal, P.Wang, CRC press, pp. 45-64, 1996.

[Tunstel 97] E. Tunstel, T. Lippincott, M. Jamshidi, "Behaviour Hierarchy for Autonomous Mobile Robots: Fuzzy Behaviour Modulation and Evolution", International Journal of Intelligent Automation and soft computing, Vol.3, No.1, pp. 37-49, 1997.

[Tunstel 97] E. Tunstel, T. Lippincott, M. Jamshidi, "Behaviour Hierarchy for Autonomous Mobile Robots: Fuzzy Behaviour Modulation and Evolution", International Journal of Intelligent Automation and soft computing, Vol.3, No.1, pp. 37-49, 1997.

To appear in: "Fusion of Soft Computing and Hard Computing for Autonomous Robotic Systems" (Eds.: Changjiu Zhou, Dario Maravall and Da Ruan), "Studies in Fuzziness and Soft Computing Series , Physica-Verlag (mid 2002)