Inhabited intelligent environments

V Callaghan, G Clarke, M Colley, H Hagras, J S Y Chin and F Doctor†

There is an increasing amount of research into the area of pervasive computing, smart homes and intelligent spaces, one example being that of the DTI-funded Pervasive Home Environment Networking (PHEN) project. Much of the current research focuses on environments populated by numerous computing devices, sensors, actuators, various wired and wireless networking systems and poses the question of how such computing environments might become 'intelligent'? Often, the proposed solution is to explicitly preprogram in the intelligence. In this paper we discuss a solution based on embedded-agents which enables emergent intelligent behaviour by predominantly implicit processes. We describe an experimental test-bed for pervasive computing, the iDorm, and report on experiments that scope the agent-learning characteristics in such environments. We also introduce a more human-directed approach to programming in pervasive environments which we refer to as task-oriented programming (TOP).

1. Introduction

In 1991 the European Community Information Society Technologies Advisory Group (ISTAG) issued a report entitled 'Scenarios for Ambient Intelligence in 2010' which became a defining statement for research into ambient intelligence (AMI) [1]. ISTAG defines ambient intelligence as an approach '... where the emphasis is on greater user friendliness, more efficient services support, user empowerment and support for human interaction. People are surrounded by intelligent intuitive interfaces, that are embedded in all kinds of objects, and an environment that is capable of recognising and responding to the presence of different individuals in a seamless, unobtrusive and often invisible way'. Such environments can even include robots [2]. In the longer term, intelligent inhabited environments are likely to be the key to mankind's successful exploration of deep space [3].

The basic AMI building block is a computer which, with the addition of a network interface, can be integrated into artefacts ranging in size from domestic appliances down to nano-scale devices [4]. These can be associated together to build both familiar and novel arrangements to generate highly personalised AMI environments. The widespread deployment of such networked devices is known as pervasive computing, a paradigm which raises numerous new scientific challenges ranging from the underlying network technology, through intelligent agents to user-interface issues. A key difference between non-networked and networked appliances is that the latter can co-ordinate their actions to create meta functions from groups of associated devices. Further, if users are given the freedom to choose these combinations of devices, then they can create unique and novel functionalities, that could not be envisaged by the manufacturers. One challenge, and the focus of much of the discussion in this paper, is how to manage and configure (program) such co-ordinated pervasive computing devices to do the end user's bidding, without the user incurring prohibitive cognitive loads — a task that, without support, could quickly become prohibitive and an obstacle to the achievement of the pervasive home-networking environment vision.

2. Degrees of intelligence and autonomy

For the AMI vision to be realised in domestic environments, people must be able to use computer-based artefacts and systems in a way that gives them some control over aspects of the system, while eliminating cognitive awareness of parts of the system they have no interest in, and are happy to leave to automation or implicit programming processes. Where the line between fully autonomous intelligent systems and manual programming should be drawn is a subject of much research (and diverse opinions). At the University of Essex we have chosen to provide an approach that allows the full spectrum of possibilities to be experimented with; we have therefore developed a range of autonomous intelligent embedded agents and some user-centric techniques. In this paper we present,

† All authors are from the University of Essex

a review of all these techniques, although we shall start by describing our test beds for intelligent spaces (iSpaces), the iDorm and the new iDorm-2.

3. The iDorm

The intelligent dormitory (iDorm) shown in Fig 1 is a real pervasive computing test-bed comprised of a large number of embedded sensors, actuators, processors and networks in the form of a student bed-sitting room. The iDorm is a multi-use, multi-user space containing areas for different activities such as sleep, work and entertaining. It contains the normal mix of furniture found in a typical student study/bedroom environment, including a bed, work desk and a wardrobe.

A common interface to the iDorm and its devices is implemented through universal plug and play (UPnP) which is an event-based communication middleware that allows devices to plug and play thus enabling automatic discovery and configuration. A gateway server is used to run the UPnP software devices that interface with the hardware devices on their respective networks. Our experimental agent mechanisms are built on top of the low-level UPnP control architecture enabling it to communicate with the UPnP devices in the iDorm and thus allowing it to monitor and control these devices. Figure 2 shows the logical network infrastructure of the iDorm.

Entertainment is one of the behaviours used as a benchmark in the iDorm for performance assessment in projects such as the BT-led Pervasive Home Environment Networking (PHEN) [5] project. There is a standard multimedia PC driving both a flat-screen monitor and a video projector which can be used for working and entertainment (see Fig 3).

Any networked computer that can run a standard Java process can access and control the iDorm directly. Thus any PC can also act as an interface to control the devices in the room. Equally interfaces to the devices could be operated from wearable artefacts that can monitor and control the iDorm wirelessly, such as a handheld PDA supporting Bluetooth wireless networking or a mobile telephone shown in Fig 4. In principle, it is possible to adjust the environment from anywhere and at any time, subject to user and device privileges. There is also an Internet fridge in the iDorm (see Fig 4(d)) that incorporates a PC with touchscreen capability, which can also be used to control the devices in the room. Control can of course still be exerted directly on the devices themselves via conventional switches, buttons, etc.

There are a variety of computers in the iDorm which are used to interface with sensors and actuators and run agents, all of them being configured as Java environments. At the low performance end we use TINI [6] and SNAP [7] embedded Internet boards; these are mainly used for sensors and actuators. There are also more powerful processor boards capable of running agents such as jStik [8] and ITX [9]. For experiments where maximum flexibility is required, it is also possible to run agents on UPnP-enabled workstations. This allows the granularity of agent to device to be varied, from an agent controlling an entire environment, down to one-to-one mappings between devices and agents.

With the success of the iDorm, Essex University is currently constructing a new test bed to support R&D in pervasive ICT. The new facility, funded by the HE SRIF programme takes the form of a domestic apartment and has been called iDorm-2.

The iDorm-2 has been built from the ground up to be an experimental pervasive computing environment with many special structural features such as cavity walls/ceilings containing power and network outlets together with provision for internal wall-based sensors and processors, etc. There are numerous networks in place ranging from wired and power-line, through wireless to broadband, and high-bandwidth multi-mode fibre connections to the outside world. All the basic

Fig 1 The iDorm.

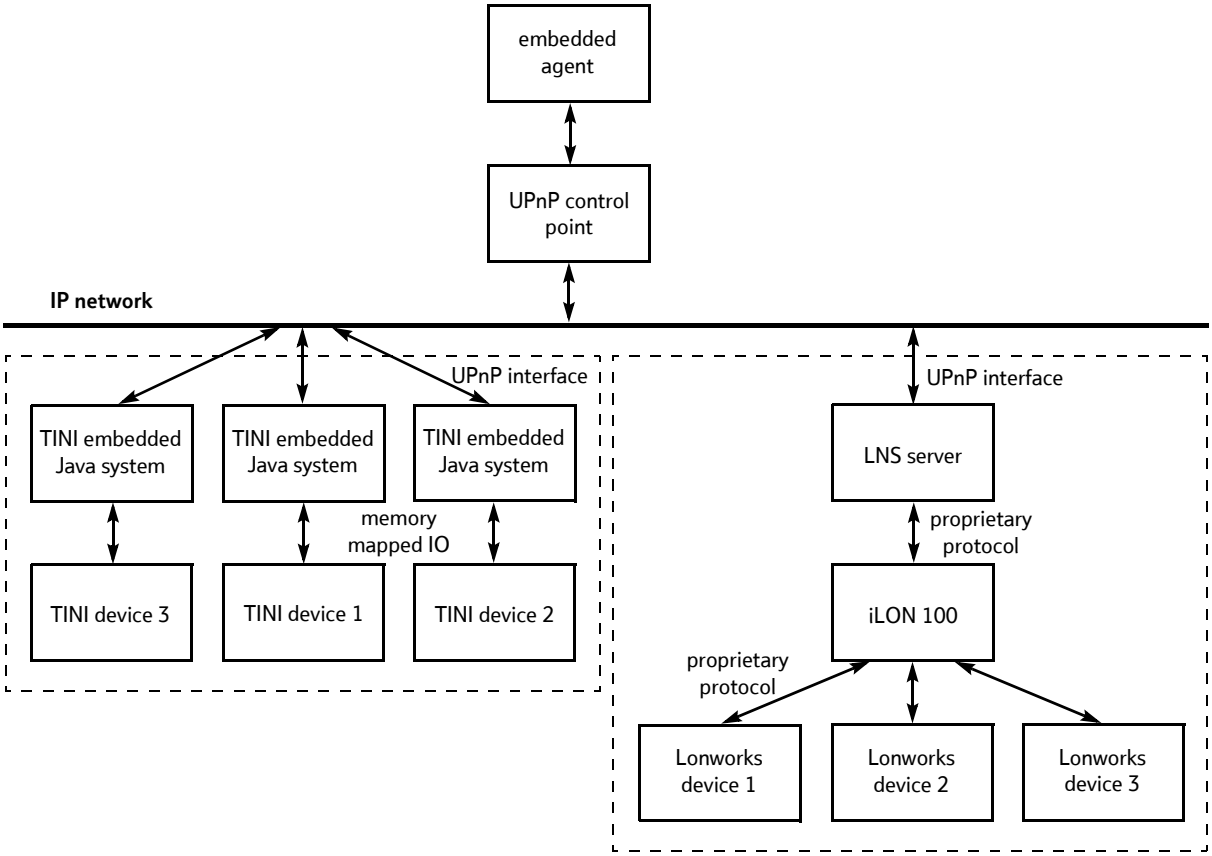

Fig 2 The iDorm logical network infrastructure.

Fig 3 Entertainment and work in the iDorm.

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

Fig 4 PC interface.

services are electrically controlled wherever possible (e.g. heating, water, doors). The basic layout of the flat is show in Fig 5 (together with a picture of its current state of build; it is due to be complete and handed over for October 2004). When finished this will be one of the few such facilities in the world.

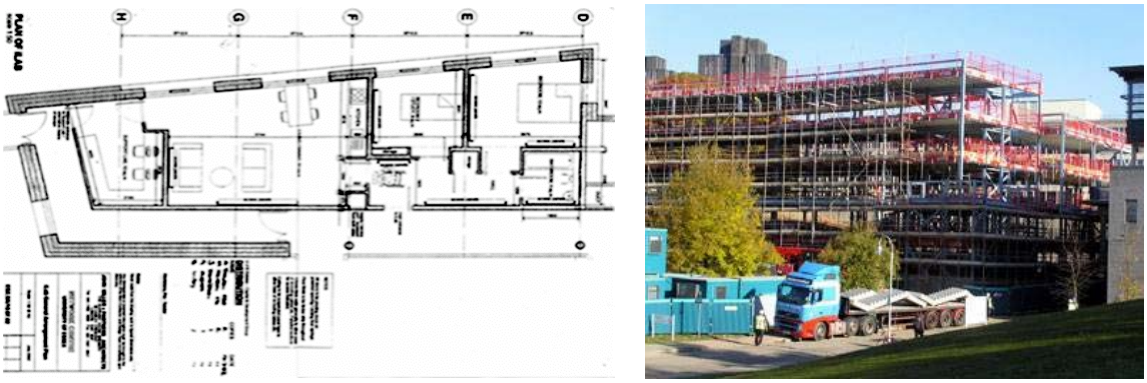

Fig 5 iDorm-2 (due to open in October 2004).

4. Embedded agents

The principal argument in support of utilising artificial intelligence (AI) in support of the creation (programming) and management (control) of intelligent pervasive computing-based spaces is that much of the cognitive load associated with using the technology (which is an obstacle to market penetration) can be off-loaded from the user to software processes. However, this is far from easy as such 'intelligent entities' operate in a computationally complex and challenging physical environment which is significantly different to that encountered in more traditional PC programming or AI. Some of the computational challenges associated with creating systems of intelligent artefacts are discussed below.

4.1 Embedded intelligence

Embedded intelligence can be regarded as the inclusion of some of the reasoning, planning and learning processes in an artefact that, if a person did it, we would regard as requiring intelligence. An intelligent artefact would normally contain only a minimal amount of 'embedded intelligence', sufficient to do the artefact task in question. Embedded computers that contain such an intelligent capability are normally referred to as 'embedded agents' [10]. Intelligent artefacts would, in effect, contain an embedded agent. Individually, such an embedded-agent can harness intelligence to undertake such tasks as enhancing device functionality (i.e. enabling the artefact to do more complex control tasks), as well as reducing configuration or programming complexity and costs by enabling the pervasive computing system to autonomously learn its own program rules, or alternatively assisting the lay end user to program rules in a non-technical way (see TOP, section 5.4).

4.2 Embedded agents and intelligent spaces

There are a variety of approaches to this problem, perhaps the most relevant being those originating from the context-aware and embedded-agent communities. In embedded-agent work the goal is to utilise some form of AI to relieve the cognitive loading associated with setting up and running an AMI system (i.e. transfer some of the cognitive processes from the person to the computer). Typically researchers have employed approaches such as neural networks, based on traditional machine-learning theory, to control the users' environment. However, these approaches utilise objective functions that either aim to derive a minimal control function that satisfies the needs of the user 'average' or are aimed at optimising between a number of competing needs (e.g. energy efficiency and user comfort). In both cases the user has little control over the system and has to accept some degree of discomfort, or adapt to the conditions determined by the AMI agents [11].

A contrasting agent-based paradigm is to see the 'user as king' and create agents that '... particularise (rather than generalise) to a specific user's needs, and respond immediately to whatever the end user demands (providing it does not violate any safety constraints)' [12, 13].

Work at Essex University (as part of the EU's Disappearing Computer Programme and the UK Government's UK-Korean Scientific Fund) has addressed this problem using behaviour-based systems [14]) and soft-computing (fuzzy logic, neural networks and genetic algorithms). This approach stems from our finding that embedded agents used in pervasive computing are equivalent to robots, experiencing similar problems with sensing, non-determinism, intractability, embodiment, etc [12]. Our earlier work [13, 15, 16] was in the field of robotics, which has allowed us to recognise the underlying similarities between robotics and intelligent artefacts. Models in both robotics and pervasive embedded computer devices have proved difficult to devise, mainly because of the intractability of the variables involved (and in the case of modelling people, non-determinism). A principal advantage of behaviour-based methods is that they discard the need for an abstract model, replacing it by the world itself, a principle most aptly summarised by Brooks as '... the world is its own best model' [14].

4.3 Agent learning

Learning can be viewed as the process of gathering information from the environment and encoding it to improve the efficiency of a system in achieving a certain goal. However, the difficulty that arises concerns finding the most appropriate learning algorithm/technique to use. Most learning algorithms use a measure of the quality of the solution, given either by examples of the desired behaviour of a system, or by an assessment of the quality of the internal and/or external state. The learning algorithm very much depends on the characteristics of the 'problem' itself. The best choice of learning algorithm can be made by comparing the problem characteristics against the learning algorithm characteristics. The following describes a limited number of these characteristics.

- Problem characteristics

  Dynamics — to what degree do the environment variables change during the learning?

  Complexity — is the set of all possible solutions, search space, finite/countable?

  Uncertainty — does the information regarding the state contain noise, and are the actions performed noisy?

  Pre-acquired knowledge — can some knowledge about the solutions be acquired before learning starts?

  Observability — is the current/past state known to the learning algorithm?

  Type of data — is the data provided discrete-valued, real-valued, and complex-structured or states and transitions?

  Feedback type — should the learning algorithm respond as an immediate, on-demand, delayed or no-response feedback?

  Physical limitations — what is the processing capability or memory size of the system where the learning algorithm runs?

- Learning algorithm characteristics

  Internal parameter type — what type of parameter does the algorithm contain and how does it change?

  Input data — what kind of input data can the learning algorithm deal with and can it adapt to noisy data?

  Solution/goal type — can the learning algorithm produce approximations in real valued functions?

  Dynamics — can the solutions be changed during the environment's execution or can the learning algorithm only change the solutions off-line?

  Parameter change — what parameters change in each phase of the learning cycle, and do they change all at the same time or only a small subset?

Another important distinction in learning agents is whether the learning is done on-line or off-line. On-line learning means that the agent performs its tasks, and can learn or adapt after each event. On-line learning is like 'on-the-job' training and places a severe requirement on the learning algorithm. It must not only be fast but also very stable and fault-tolerant. Other hotly debated issues are whether supervised or unsupervised learning is best.

Later we present the ISL and the AOFIS as examples of the unsupervised agent. The general challenges faced by designers of embedded agents for such environment, were discussed at a recent workshop on Ubiquitous Computing in Domestic Environments [13].

4.4 Application-level emergent behaviour

In pervasive computing systems, the embedded-agent host (frequently an appliance) has a network connection allowing the agents to have a view of their neighbours, thereby facilitating co-ordinated actions from groups of embedded agents. The key difference to isolated appliances is that those participating in groups not only have their individual functionality (as designed by the manufacturer), but they also assume a group functionality that can be something that was not envisaged by the manufacturers. In fact, if there are only weak constraints on association of appliances, it is possible for the user to program unique co-ordinated actions (i.e. unique collective functionality) that were not envisaged by the different manufacturers offering the component appliances. This enables an application-level emergent behaviour or functionality (something that while enabled by the system, was not specified by the system). This naturally gives rise to questions such as the balance between pre-specified functionality and emergent functionality, and what or who is responsible for the association between devices and the programming of the basic behaviours.

Later in this paper we discuss various approaches to this challenge. Task-oriented programming (TOP) provides an explicit means of directly harnessing user creativity to generate emergent applications, while incremental synchronous learning (ISL) and the adaptive on-line fuzzy inference system (AOFIS) involve various degrees of user interaction, using both supervised and unsupervised learning paradigms to generate emergent application-level functionality.

4.5 Machine-level emergent behaviour

In the behaviour based approach to AI, the equivalent to reasoning and planning in traditional AI is produced by arranging for an agent to have a number of competing processes that are vying for control of the agent. The 'sensory context' determines the degree to which any process influences the agent. Thus, as sensing is derived from what is effectively a non-deterministic world, the solutions from this process are equally non-deterministic and result in what is termed 'emergent behaviour' (behaviours or solutions that emerged but were not explicitly programmed). Anything that affects the context can thus have a hand in this machine-level emergent behaviour. For example, the connections (associations) between devices critically affect the sensed data. Thus agent-driven associations, or user-driven associations, will be closely associated with emergent behaviour. Emergent behaviour is also sometimes described as emergent solutions. The freedom to make *ad hoc* associations is an important factor in this process, as without them it is difficult to see how emergent functionality could be achieved. At the University of Essex we are researching into what we term promiscuous association — the freedom for agents to form their own associations in as open a way as possible. This approach opens up the possibility of using formally specified ontologies of devices and groups of devices. It is important to understand that being autonomous and promiscuous (open to making associations with other artefacts) does not imply undirected or unsafe behaviour. Agents can have basic fixed rules built into them that prevent them taking specified actions deemed unsafe.

4.6 Multi-agents

Our underlying paradigm for all agents is that they are associated with actuators (they are essentially control agents rather than information-processing agents). In the underlying agent model, multi-agent operation is supported via three modes. In the first, sensory and actuator parameters are simply made available to other agents. In the second mode, agents make a 'compressed' version of this information (or their internal state) available to the wider network. In a behaviour-based agent, such as the ISL, the compressed data takes the form of which behaviours are active (and to what degree). The general philosophy we have adopted is that data from remote agents is simply treated in the same way as all other sensor data. 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, which regards remote agents as simply more sensors (albeit more sophisticated sensors). We have found that receiving high-level processed information from remote agents, such as 'the iDorm is occupied' is more useful than being given the low-level sensor information from the remote agent that gave rise to this higher-level characterisation. This compressed form both relieves agent processing overheads and reduces network loading. A third approach we have developed is the use of inter-agent communication languages. Standardised agent communication languages (e.g. KQML and FIPA) tend to be too big to use on embedded computers (many tens of megabytes) and are not well matched in terms of functionality to them. We have generated research that has looked at the problem of developing a lightweight agent communication language and the interested reader is referred to our description of the Distributed Intelligent Building Agent Language (DIBAL) [17]. Finally, in the home environment (rather than a general unconstrained pervasive environment), because the number of connected appliances is relatively tractable (no more than a few hundred), a widely adopted approach at a network level is to fully connect all the appliances, relegating the issue of what appliance will collaborate with any other to the application level. This approach has been successfully applied at the University of Essex [18, 19].

4.7 Knowledge in rule-based agents

One reason we have opted for fuzzy logic rather than neural networks is that the knowledge acquired by the agent is gathered in human linguistic terms. A typical rules set from the iDorm is presented in Fig 6. It is made up of simple, if somewhat large *IF THEN ELSE* rule sets. Such rules are intrinsically well structured as they are based on mathematical logic sets. Meta structures can also be used. For example, at the meta level, rule sets can also be characterised according to context such as rule sets for Mr A relating to Context B (e.g. a bedroom). Thus, from such rule sets it is possible to perform meta functions such as deriving the closest rule set for a new user — Ms C — based upon rule sets from other users in the same context.

IF **InternalLightLevel** *is* **VVLOW** *AND* **ExternalLightLevel** *is* **VVLOW** *AND* **InternalTemperature** *is* **VVHIGH** *AND* **ExternalTemperature** *is* **MEDIUM** *AND* **ChairPressure** *is* **OFF** *AND* **BedPressure** *is* **ON** *AND* **Hour** *is* **Evening** *THEN* **ACTION_LIGHT1_value** *is* **VHIGH** *AND* **ACTION_Light2_value** *is* **HIGH** *AND* **ACTION_LIGHT3_value** *is* **LOW** *AND* **ACTION_Light4_value** *is* **VVLOW** *AND* **ACTION_Blind_state** *is* **CLOSED** *AND* **ACTION_Bedlight_state** *is* **ON** *AND* **ACTION_DeskLight_state** *is* **OFF** *AND* **ACTION_Heater_state** *is* **OFF** *AND* **ACTION_MSWord_state** *is* **STOPPED** *AND* **ACTION_MSMediaPlayer_state** *is* **RUNNING**

Fig 6 Example of rule representation.

5. Examples of agents

We have developed a number of agents that can deal with the problems discussed above. The main approaches we have developed are based on fuzzy logic. Fuzzy logic is particularly appropriate as it can describe inexact (and analogue) parameters using human-readable linguistic rules, offering a framework for representing imprecise and uncertain knowledge. Thus

it is well suited for developing control on the basis of inexact sensing and actuation which, when coupled to behaviour-based agent architectures, can deal with the non-determinism which sometimes characterises human behaviour. We believe this has similarities to the way people make decisions as it uses a mode of approximate reasoning, which allows it to deal with vague and incomplete information. We have shown that fuzzy logic can be applied well to the pervasive computing environment [20—22] such as the iDorm [23, 24] and have developed and tested two fuzzy-based embedded agents in the iDorm, namely the incremental synchronous learning agent [25] and the adaptive on-line fuzzy inference system agent [26—28]. These agents have been run on commercial and in-house produced hardware. The photograph in Fig 7 shows a hardware networked agent platform produced at the University of Essex and used to manage the iDorm pervasive computing community.

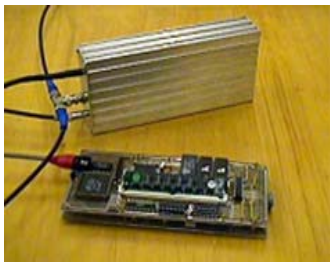

Fig 7 Agent prototype.

5.1 The incremental synchronous learning agent

In general terms, the ISL embedded-agent work is broadly situated within the behaviour based architecture work pioneered by Brooks at MIT, consisting of many simple co-operating sub-control units. Our approach differs to other work in that we use fuzzy logic based sub-control units, arranging them in a hierarchy (see Fig 8) and employing a user-driven technique to learn the fuzzy rules on-line and in real time. It is well known that it is often difficult to determine parameters for fuzzy systems. In most fuzzy systems, the fuzzy rules were determined and tuned through trial and error by human operators. It normally takes many iterations to determine and tune them. As the number of input variables increases (iSpace agents develop large numbers of rules due to particularisation), the number of rules increases disproportionately, which can cause difficulty in matching and choosing between large numbers of rules. Thus the introduction of a mechanism to learn fuzzy rules was a significant advance. In the ISL agent we implement each behaviour as a fuzzy process and then use a higher level fuzzy process to co-ordinate them. The resultant architecture takes the form of a hierarchical tree structure (as shown in Fig 8). This approach has the following technical advantages:

- it simplifies the design of the embedded agent, reducing the number of rules to be determined (in previous work we have given examples of rules reduction of two orders of magnitude via the use of hierarchies),
- it uses the benefits of fuzzy logic to deal with imprecision and uncertainty,
- it provides a flexible structure where new behaviours can be added (e.g. comfort behaviours) or modified easily,
- it utilises a continuous activation scheme for behaviour co-ordination which provides a smoother response than switched schema.

The learning process involves the creation of user behaviours. This is done interactively using reinforcement where the controller takes actions and monitors these actions to see if they satisfy the user or not, until a degree of satisfaction is achieved. The behaviours, resident inside the agent, take their input from sensors and appliances and adjust effector and appliance outputs (according to predetermined, but settable, levels). The complexities of learning and negotiating satisfactory values for multiple users would depend upon having a reliable means of identifying different users.

It is clear that, in order for an appliance-based agent to autonomously particularise its service to an individual, some form of learning is essential [13]. In the ISL, learning takes the form of adapting the 'usage' behaviour rule base, according to the user's actions. To do this we utilise an evolutionary computing mechanism based on a novel hierarchical genetic algorithm (GA) technique which modifies the fuzzy controller rule-sets through interaction with the environment and user.

The hub of the GA learning architecture is what we refer to as an Associative Experience Engine [29]. Briefly, each behaviour is a fuzzy logic controller (FLC) that has two parameters that can be modified — a rule base (RB) and its associated membership functions (MFs). In our learning we modify the rule base. The architecture, as adapted for pervasive computing embedded agents, is shown in Fig 8. The behaviours receive their inputs from sensors and provide outputs to the actuators via the co-ordinator that weights their effect. When the system fails to have the desired response (e.g. an occupant manually changes an effector setting), the learning cycle begins.

When a learning cycle is initiated, the most active behaviour (i.e. that most responsible for the agent behaviour) is provided to the learning focus from the co-ordinator (the fuzzy engine which weights contributions to the outputs), which uses the information to point at the rule set to be modified (i.e. learnt) or exchanged. Initially, the contextual prompter (which gets a

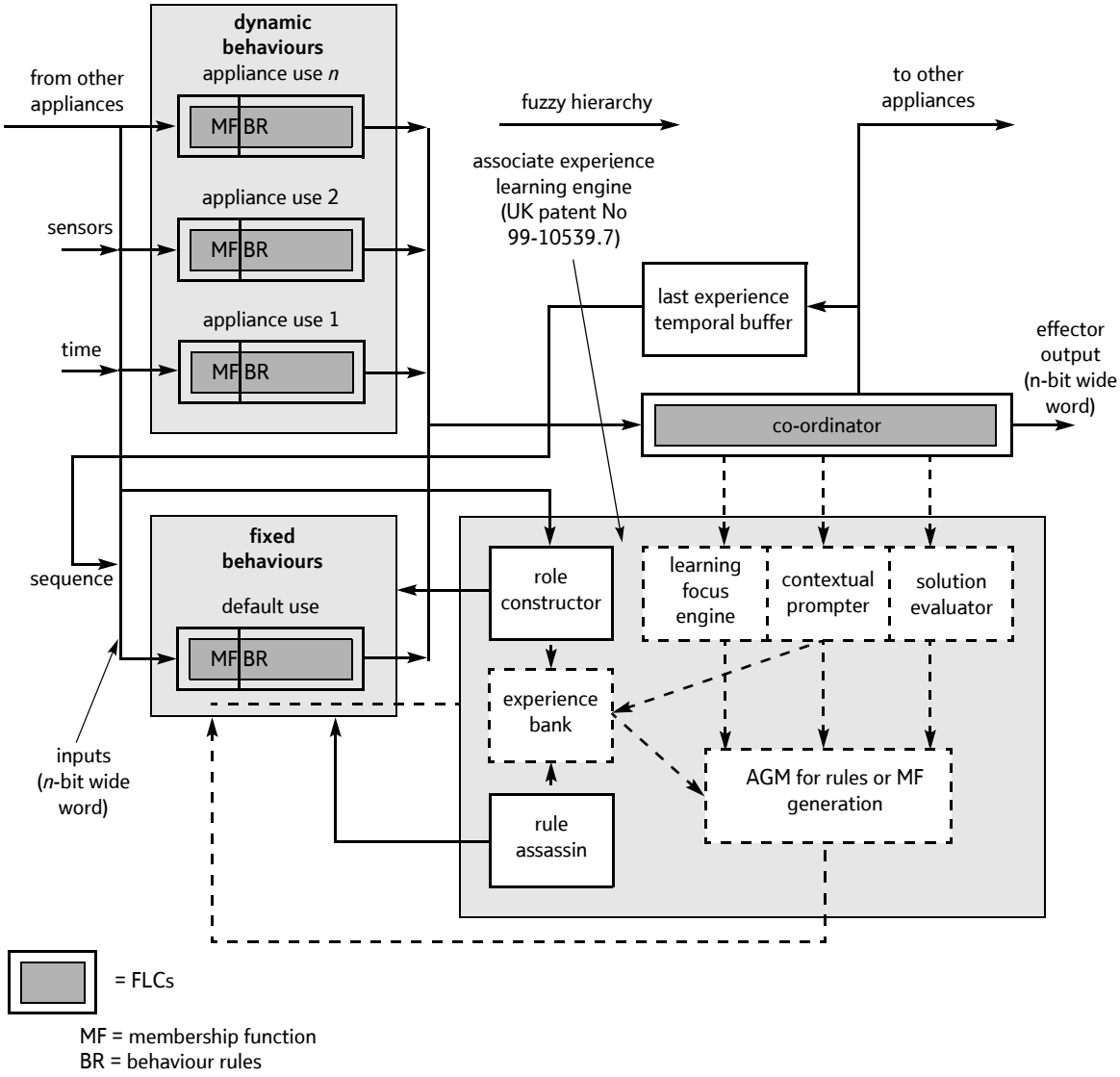

Fig 8 ISL Embedded-agent architecture.

characterisation of the situation, an experience, from the co-ordinator) is used to make comparison to see whether there is a suitable behaviour rule set in the experience bank. If there is a suitable experience, it is used. When the past experiences do not satisfy the occupant's needs, we use the best-fit experiences to reduce the search space by pointing to a better starting point, which is the experience with the largest fitness. We then fire an adaptive genetic mechanism (AGM) using adaptive learning parameters to speed the search for new solutions. The AGM is constrained to produce new solutions in a certain range defined by the contextual prompter to avoid the AGM searching options where solutions are unlikely to be found. By using these mechanisms we narrow the AGM search space massively, thus improving its efficiency. After generating a new solution, the system tests it and gives it fitness through the solution evaluator. The AGM provides new options via operators such as crossover and mutation until a satisfactory solution is achieved.

The system then remains with this set of active rules (an experience) until the user's behaviour indicates a change of preference (e.g. has developed a new habit), signalled by a manual change to one of the effectors, when the learning process described above is repeated. In the case of a new occupant in the room the contextual prompter gets and activates the most suitable rule base from the experience bank or if this proves unsuitable the system re-starts the learning cycle above. The solution evaluator assigns a fitness value to each rule base stored in the experience bank. When the experience bank is full, we have to delete some experiences.

To assist with this the rule assassin determines which rules are removed according to their importance (as set by the solution evaluator). The last experience temporal buffer feeds back to the inputs a compressed form of the $n-1$ state, thereby providing a mechanism to deal with temporal sequences.

5.2 *Adaptive on-line fuzzy inference system agent*

Like the ISL agent, AOFIS is based on fuzzy logic. We utilise an unsupervised data-driven one-pass approach for extracting fuzzy rules and membership functions from data to learn a fuzzy logic controller (FLC) that will model the user's behaviours when using iDorm-based devices. It differs from the ISL in that it not only learns controller rules, but it also learns membership functions (a significant advance on the ISL which has fixed membership functions). The data is collected by monitoring the user's use of the iDorm over a period of time. The learnt FLC provides an inference mechanism that produces output control responses based on the current state of the inputs. The AOFIS adaptive FLC will therefore control a pervasive computing community, such as the iDorm, on behalf of the user and will also allow the rules to be adapted on-line as the user's behaviour changes over time. This approach aims to realise the vision of AMI and support the aims of pervasive computing in the following ways:

- the agent is responsive to the particular needs and preferences of the user,
- the user is always in control and can override the agent at any time,
- the agent learns and controls its environment in a non-intrusive way (although users may be aware of the high-tech interface, they are unaware of the agent's presence),
- the agent uses a simple one-pass learning mechanism for learning the user's behaviours, and thus it is not computationally expensive,
- the agent's learnt behaviours can be adapted on-line as a result of changes in the user's behaviour,
- learning is life-long in that agent behaviours can be adapted and extended over a long period of time as a result of changes in the pervasive computing environment.

AOFIS involves five phases — monitoring the user's interactions and capturing input/output data associated with their actions, extraction of the fuzzy membership functions from the data, extraction of the fuzzy rules from the recorded data, the agent control, and the life long learning and adaptation mechanism. The last two phases are control loops that once initiated receive inputs as either monitored sensor changes that produce appropriate output control responses based on the set of learnt rules, or user action requests that cause the learnt rules to be adapted before an appropriate output control response is produced. These five phases are illustrated in Fig 9.

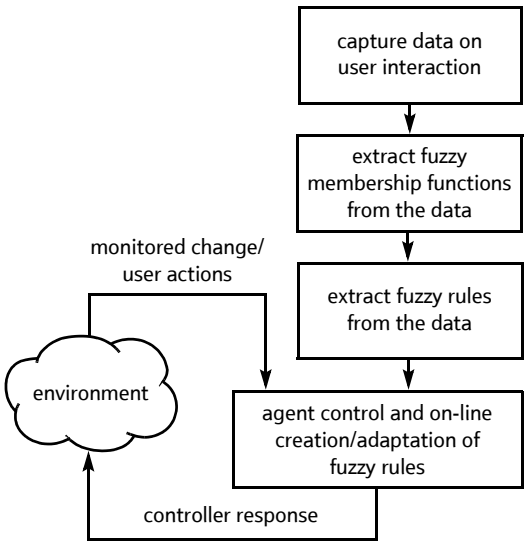

Fig 9 Phases of AOFIS.

The agent initially monitors the user's actions in the environment. Whenever the user changes actuator settings, the agent records a 'snapshot' of the current inputs (sensor states) and the outputs (actuator states with the new values of whichever actuators were adjusted by the user). These 'snapshots' are accumulated over a period of time so that the agent observes as much of the user's interactions within the environment as possible. AOFIS learns a descriptive model of the user's behaviours from the data accumulated by the agent. In our experiments in the iDorm we used seven sensors for our inputs and ten actuators for our outputs with a user spending up to three days in the iDorm. The fuzzy rules which are extracted represent local models that map a set of inputs to the set of outputs without the need for formulating any mathematical model. Individual rules can therefore be adapted on-line influencing only specific parts of the descriptive model learnt by the agent.

It is necessary to be able to categorise the accumulated user input/output data into a set of fuzzy membership functions which quantify the raw crisp values of the sensors and actuators into linguistic labels. AOFIS is based on learning the particularised behaviours of the user and therefore requires these membership functions be defined from the user's input/output data recorded by the agent. A double clustering approach combining Fuzzy-C-Means (FCM) and hierarchical clustering, is used for extracting fuzzy membership functions from the user data. This is a simple and effective approach where the objective is to build models at a certain level of information granularity that can be quantified in terms of fuzzy sets.

Once the agent has extracted the membership functions and the set of rules from the user input/output

data, it has then learnt the FLC that captures the human behaviour. The agent FLC can start controlling the pervasive computing community on behalf of the user. The agent starts to monitor the state of the pervasive community and affect actuators based on its learnt FLC that approximate the particularised preferences of the user. Figure 10 illustrates the FLC which consists of a fuzzifier, rule base, fuzzy inference engine and defuzzifier.

In conformity with the non-intrusive aspect of intelligence [26—28], whenever users are not happy with the behaviour of the pervasive computing device or community, they can always override the agent's control responses by simply altering the manual control of the system. When this occurs the agent will adapt its rules on-line or add new rules based on the new user preferences. This process incorporates what we term 'learning inertia' where the agent delays adapting its learnt rules until the user preference for changing a particular set of actuator values has reoccurred a number of times. This prevents the agent adapting its rules in response to 'one-off' user actions that do not reflect a marked change in the user's habitual behaviour (this 'learning inertia' parameter is user settable). As rules are adapted it is sensible to preserve old rules so that they can be recalled by the agent in the future if they are more appropriate than the current rules. Whenever the user overrides the agent's control outputs and overrides any of the controlled output devices, a snapshot of the state of the environment is recorded and passed to the rule-adaptation routine. The AOFIS agent supports the notion of life-long learning in that it adapts its rules as the state of the pervasive community and the user preferences vary over a significantly long period of time. Due to the flexibility of AOFIS, the initially learnt FLC can be easily extended to both adapt existing rules, as well as adding new rules. The fuzzy nature of the rules permits them to capture a wide range values for each input and output parameter. This allows the rules to continue to operate even if there is a gradual change in the environment. If, however, there is a significant change in the environment or the user's activity is no longer captured by the existing rules, the agent will automatically create new rules that satisfy the current conditions. The agent will therefore unobtrusively and incrementally extend its behaviours which can then be adapted to satisfy a pervasive device and community user.

5.3 Benchmarking and comparative performance

We have also implemented other soft computing agents namely genetic programming (GP), the adaptive-neuro fuzzy inference system (ANFIS) and the multilayer perceptron neural network. The data set obtained from the iDorm (see Fig 11) during the AOFIS monitoring phase comprised of 408 instances and was randomised into six samples. Each sample was then split into a training and test set consisting of 272 and 136 instances respectively. The performance error for each technique was obtained on the test instances as the root mean squared error which was also scaled to account for the different ranges of the output parameters. The GP used a population of 200 individuals evolving them over 200 generations. The GP evolved both the rules and the

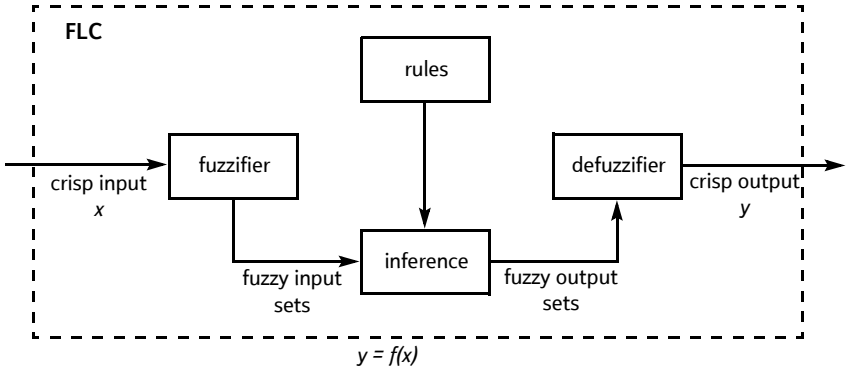

Fig 10 AOFIS FLC.

Fig 11 User gathering experimental data in the iDorm.

fuzzy sets. Each individual was represented as a tree composed of 'and' and 'or' operators as the internal nodes, and triangular and trapezoidal membership functions as terminal nodes. The parameters of the membership functions were also evolved in parallel with the structure. The search started with a randomly generated set of rules and parameters, which were then optimised by means of genetic operators. The GP-based approach for optimising an FLC was tested with different numbers of fuzzy sets. In ANFIS subtractive clustering was used to generate an initial TSK-type fuzzy inference system. Back propagation was used to learn the premise parameters, while least square estimation was used to determine the consequent parameters. An iteration of the learning procedure consisted of two parts where the first part propagated the input patterns and estimated optimal consequent parameters through an iterative least squares procedure. The second part used back propagation to modify the antecedent membership functions. We tested ANFIS with a range of different cluster radii values. The multilayer perceptron (MLP) back-propagation neural network was tested with different numbers of hidden nodes in a single hidden layer. We tested the AOFIS with different numbers of fuzzy sets and the membership function overlap threshold was set to 0.5 as this gave both a sufficient degree of overlap while allowing the system to distinguish between the ranges covered by each fuzzy set. Tables 1 and 2 illustrate the scaled root mean squared error (RMSE) and scaled standard deviation (STD) for each technique averaged over the six randomised samples, and corresponding to the values of the variable parameter tested for each approach.

The results above show that the optimum number of fuzzy sets for AOFIS was 7 and on average AOFIS produced 186 rules. The GP in comparison gave a marginally lower error for 7 fuzzy sets. Both ANFIS and the MLP on average gave a higher error than AOFIS. The ANFIS only learns multi-input-single-output (MISO) FLC and had to be run repeatedly for each output parameter. The FLC produced was therefore only representative of a MISO system. Another restriction with ANFIS was that it generates TSK FLCs, where the consequent parameters are represented as either linear or constant values, rather than linguistic variables as is the case with Mamdani FLCs. These linguistic variables are very important to understanding the human behaviour. It should be noted that the AOFIS generates multi-input, multi-output (MIMO) Mamdani FLCs representing rules in a more descriptive human-readable form which is advantageous for pervasive computing communities or other ambient intelligent systems, as they deal with people whose behaviours are more easily described in such linguistic terms. The iterative nature of the GP makes it highly computationally intensive and this also applies to both ANFIS and the MLP which are also iterative-based approaches. AOFIS is far less computationally intensive due to the one-pass procedure it employs, and is therefore more favourable for an embedded agent. Neither ANFIS nor the GP-based approach can easily be adapted on-line as this would require their internal structures to be re-learnt if either new rules were to be added or existing rules were adapted. So the AOFIS method is unique in that it can learn a good model of the user's behaviour which can then be adapted on-line in a life-long mode, and in a non-intrusive manner, unlike other methods which need to repeat a time-consuming learning cycle to adapt the user's behaviour. Hence, in summary, the AOFIS agent proved to be the best for on-line learning and adaptation, moreover it is was computationally less intensive and better suited to on-line learning than the other approaches compared. Finally, at the outset of our work it was not clear how long (if at all) it would take for such learning in this type of environment to reach a steady state. Our initial results (see Fig 12) indicate this is possible within a day although we would need to conduct experiences over much longer periods to catch other cycles, such as annual climate-related variations.

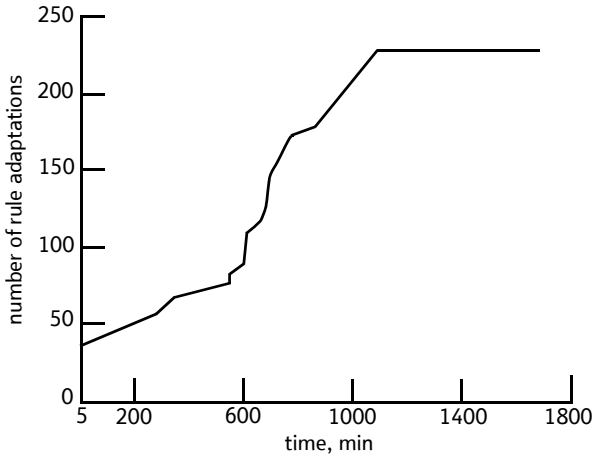

Fig 12 Typical learning rate of FLC-based agent.

5.4 *Task-oriented programming*

Finally, while autonomous agents may appeal to many people, their acceptance is not universal. Some lay people distrust autonomous agents and prefer to exercise direct control over what is being learnt and when (particularly when it is in the private space of their home). Moreover, there are other reasons advanced in support of a more human-driven involvement, such as exploiting the creative talents of people by providing them with the means to become designers of their own systems. To explore this aspect of our inhabited intelligent environment work we have recently opened up a complementary strand of research which we refer to as task-oriented programming. It is based on an

Table 1 Average scaled RMSE.

Average scaled root mean squared error (SRMSE) for six randomised samples of the dataset							
AOFIS		GA		ANFIS		MLP	
Number of fuzzy sets	SRMSE	Number of fuzzy sets	SRMSE	Cluster radii	SRMSE	Number of hidden nodes	SRMSE
2	0.2148	2	0.1235	0.3	1.3269	2	0.2129
3	0.1476	3	0.1156	0.4	0.9229	4	0.1718
4	0.1461	4	0.1189	0.5	0.2582	6	0.1732
5	0.1364	5	0.1106	0.6	0.1661	8	0.1571
6	0.1352	6	0.1210	0.7	0.1669	10	0.1555
7	0.1261	7	0.1193	0.8	0.1418	20	0.1621
8	0.1326	8	0.1173	0.9	0.1213	30	0.1705
9	0.1472	9	0.1202	1.0	0.1157	40	0.1667
10	0.1537	10	0.1235	1.1	0.1201	50	0.1768
11	0.1696	11	0.1110	1.2	0.1168	60	0.1711
12	0.1999	12	0.1201	1.3	0.1131	70	0.1712
13	0.2246	13	0.1169	1.4	0.1131	80	0.1770
14	0.2337	14	0.1120	1.5	0.1118	90	0.1767
15	0.2460	15	0.1089	1.6	0.1130	100	0.1924
16	0.2459	16	0.1225	1.7	0.1115	200	0.2027
17	0.2732	17	0.1146	1.8	0.1137	300	0.2258
18	0.2747	18	0.1188	1.9	0.1182	400	0.2365
19	0.2771	19	0.1159	2.0	0.1189	500	0.2424
20	0.2839	20	0.1143				

Table 2 Average scaled STD.

Average scaled standard deviation (SSTD) for six randomised samples of the dataset							
AOFIS		GA		ANFIS		MLP	
Number of fuzzy sets	SSTD	Number of fuzzy sets	SSTD	Cluster radii	SSTD	Number of hidden nodes	SSTD
2	0.1896	2	0.1128	0.3	1.2839	2	0.1499
3	0.1350	3	0.1063	0.4	0.9001	4	0.1299
4	0.1354	4	0.1094	0.5	0.2440	6	0.1277
5	0.1277	5	0.1026	0.6	0.1522	8	0.1193
6	0.1280	6	0.1121	0.7	0.1518	10	0.1160
7	0.1200	7	0.1107	0.8	0.1257	20	0.1198
8	0.1266	8	0.1085	0.9	0.1038	30	0.1229
9	0.1409	9	0.1117	1.0	0.0972	40	0.1234
10	0.1472	10	0.1145	1.1	0.1007	50	0.1245
11	0.1626	11	0.1026	1.2	0.0961	60	0.1234
12	0.1912	12	0.1115	1.3	0.0920	70	0.1222
13	0.2133	13	0.1084	1.4	0.0924	80	0.1283
14	0.2218	14	0.1031	1.5	0.0906	90	0.1272
15	0.2323	15	0.1007	1.6	0.0911	100	0.1333
16	0.2318	16	0.1128	1.7	0.0891	200	0.1366
17	0.2557	17	0.1063	1.8	0.0909	300	0.1503
18	0.2568	18	0.1090	1.9	0.0951	400	0.1674
19	0.2588	19	0.1075	2.0	0.0937	500	0.1676
20	0.2646	20	0.1051				

approach that puts the user at the centre of the system-programming experience by exchanging autonomous learning for taught user-driven supervision. In this approach a user defines a community of co-ordinating pervasive devices and 'programs by example' the system usage rules and co-ordinating actions for groups of pervasive devices. Both TOP and our autonomous agent approaches are based around the creation of linguistic (human-readable) rule sets (generally of the form of *IF THEN ELSE* rules). These rule sets are effectively the 'program code' which, in our TOP paradigm, are generated when a user demonstrates their desired tasks to the system in an explicit 'teaching session'. These rule sets are then interpreted by a TOP engine. Figure 13

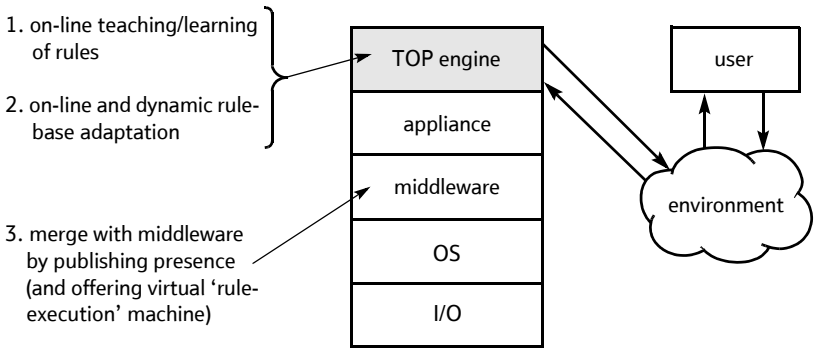

Fig 13 The three phases of TOP.

illustrates a flow chart of the main TOP phases. Clearly, there are more complex issues involved than this brief introduction can discuss (that will be the focus of later papers), but the intention of including this section is simply to indicate one future direction of our work. We are investigating this approach as part of the PHEN project, where we hope to contrast this user-centric (explicit) approach with our more (implicit) autonomous learning paradigms.

6. Conclusions and future directions

Both the ISL and AOFIS provide life-long learning and adaptation for pervasive devices and communities. Both techniques were evaluated by arranging for users to live in and use the iDorm for periods of up to five days. Both techniques performed well in handling human behaviour (with all the uncertainties involved), and in dealing with complex sensors, actuators and control. The agents operated in a non-intrusive manner allowing the user to continue operating the pervasive computing device or community in a normal way, while the agents learn controllers that satisfy the user's required behaviour. In contrast, the TOP approach deliberately seeks to involve the user in the learning phase, providing explicit control of what and when the agent learns. Contrasting these two approaches will allow us to evaluate the arguments for and against increased agent autonomy. In all the approaches, the underlying science is based on methods that are practical to implement in embedded computers.

Our work is taking a number of directions. Firstly, we are continuing to try to develop and experiment with new types of autonomous intelligent embedded agents. For example, we have projects under way looking at new type-2 fuzzy logic-based agents and new types of neuro-fuzzy agents. We are also mindful of the role that mood and emotions play in making decisions and have begun a project that is seeking to enrich the decision space of agents by adding sensed data on emotions. We have also embarked on two projects concerned with investigating the development of agents at a nano-scale; one project is looking at nano agents in fluids, the other as part of smart surfaces. Finally, to gather more realistic and meaningful results for all our research into pervasive environments, we need better data and so with SRIF support we have embarked on the construction of a new purpose-built test bed (called the iDorm-2) for pervasive computing and ambient intelligence work . The iDorm-2 is a full size domestic flat built from scratch to facilitate experimentation with pervasive computing technology. Apart from being equipped with the latest pervasive computing appliances, and having been constructed to facilitate easy experimentation, the major advantage of the iDorm-2 is that we will be able to get much longer periods of experimentation as people will be able to stay in the environment for weeks and months. Thus we look forward to being able to report more interesting and useful results when this facility comes on line in October 2004.

Acknowledgements

We are pleased to acknowledge support from the EU's Disappearing Computer programme, the UK-Korean Scientific Fund and the DTI's Next Wave Technology programme whose generous support has enabled this work. We should also like to express our gratitude to Phil Bull and Rowan Limb of BT Exact for their encouragement and support in writing this paper.

References

1 IST Advisory Group: 'Scenarios for ambient intelligence in 2010', Final report (February 2001).

2 Colley M, Clarke G, Hagras H and Callaghan V: 'Intelligent inhabited environments: co-operative robotics and buildings', 32nd International Symposium on Robotics (ISR 2001), Seoul, Korea (April 2001).

3 Clarke G, Callaghan V and Pounds-Cornish A: 'Intelligent habitats and the future: the interaction of people, agents and environmental artefacts', 4S/EASST Conference on Technoscience, Citizenship and Culture in the 21st Century, Vienna (September 2000).

4 Chin J S Y and Callaghan V: 'Embedded-Internet devices: a means of realizing the pervasive computing vision', IADIS International Conference, Algarve, Portugal (November 2003).

5 PHEN — http://iieg.essex.ac.uk/phen

6 Tiny Internet Interface — http://www.ibutton.com/TINI/

7 Imsys Technologies — http://www.imsys.se/

8 JStik — http://jstik.systronix.com/

9 Mini-itx — http://www.mini-itx.com/

10 Callaghan V, Clarke G and Pounds-Cornish A: 'Buildings as intelligent autonomous systems: a model for integrating personal and building agents', 6th International Conference on Intelligent Autonomous Systems (IAS-6), Venice, Italy (July 2000).

11 Mozer M: 'The neural network house: an environment that adapts to its inhabitants', in Proceedings of American Association for Artificial Intelligence, Spring Symposium on Intelligent Environments, AAAI Press, pp 110—114 (1998).

12 Callaghan V, Clarke G, Colley M and Hagras H: 'A soft-computing DAI architecture for intelligent buildings', Journal of Studies in Fuzziness and Soft Computing on Soft Computing Agents, Physica, Springer Verlag (June 2001).

13 Callaghan V, Colley M, Clarke G and Hagras H: 'Embedded intelligence: research issues for ubiquitous computing', Proceedings of the Ubiquitous Computing in Domestic Environments Conference, Nottingham (September 2001).

14 Brooks R: 'Intelligence without representation', Artificial Intelligence, 47, pp 139—159 (1991).

15 Hagras H, Callaghan V, Colley M and Clarke G: 'A hierarchical fuzzy genetic agent architecture for intelligent buildings sensing and control', International Conference on Recent Advances in Soft Computing (RASC 2000), Leicester, UK (June 2000).

16 Hagras H, Callaghan V, Colley M and Clarke G: 'A hierarchical fuzzy genetic multi-agent architecture for intelligent buildings learning, adaptation and control', International Journal of Information Sciences (August 2001).

17 Cayci F, Callaghan V and Clarke G: 'DIBAL — a distributed intelligent building agent language', 6th International Conference on Information Systems Analysis and Synthesis (ISAS 2000), Orlando, Florida (July 2000).

18 Duman H, Hagras H A K and Callaghan V: 'A soft-computing based approach to intelligent association in agent-based ambient-intelligence environment', 4th International Conference on Recent Advances in Soft Computing, Nottingham, United Kingdom (2002).

19 Duman H, Hagras H A K, Callaghan V, Clarke G S and Colley M J: 'Intelligent association in agent-based ubiquitous computing environments', International Conference on Control, Automation, and Systems, Muju, Korea (2002).

20 Hagras H A K, Callaghan V, Clarke G S, Colley M J, Pounds-Cornish A, Holmes A and Duman H: 'Incremental synchronous learning for embedded-agents operating in ubiquitous computing environments', in Soft Computing Agents: A New Perspective for Dynamic Information Systems, IOS Press (2002).

21 Hagras H A K, Colley M J, Callaghan V, Clarke G S and Duman H: 'A fuzzy incremental synchronous learning technique for embedded agents learning and control in intelligent inhabited environments', Proceedings of the 2002 IEEE International Conference on Fuzzy systems, Hawaii pp 139— 145 (2002).

22 Hagras H A K, Callaghan V, Colley M J, Clarke G S and Duman H: 'Online learning and adaptation for intelligent embedded agents operating in domestic environments', in Maravall D and Zhou D A C (Eds): 'Fusion of soft computing and hard computing for autonomous robotic systems', Physica, Springer Verlag (2002).

23 Holmes A, Duman H and Pounds-Cornish A: 'The iDORM: gateway to heterogeneous networking environments', International ITEA Workshop on Virtual Home Environments, Paderborn, Germany (February 2002).

24 Pounds-Cornish A and Holmes A: 'The iDorm — a practical deployment of grid technology', 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany (May 2002).

25 Hagras H A K, Callaghan V, Colley M J, Clarke G S and Duman H: 'A fuzzy logic based embedded agent approach to ambient intelligence in ubiquitous computing environments', IEEE Intelligent Systems Journal (2004).

26 Doctor F, Hagras H A K and Callaghan V: 'An intelligent fuzzy agent approach for realising ambient intelligence in intelligent inhabited environments', to appear in IEEE Trans on Systems, Man and Cybernetics, Part A: Systems and Humans, Special Issue on Ambient Intelligence.

27 Doctor F, Hagras H and Callaghan V: 'A type-2 fuzzy embedded agent for ubiquitous computing environments', to appear in Proc IEEE International Conference on Fuzzy systems, Budapest, Hungary (2004).

28 Doctor F, Hagras H and Callaghan V: 'An adaptive fuzzy learning mechanism for intelligent agents in ubiquitous computing environments', to appear in Proc Worldwide Automation Congress, Symposium on Intelligent Automation and Control Seville, Spain (2004).

29 'Genetic-Fuzzy Controller', UK Patent No 99 10539.7, 7th May 1999.

Victor Callaghan BEng, PhD, CEng, MBCS, MIEE, holds a PhD in Computing and BEng in Electronic Engineering from the University of Sheffield. His main expertise concerns embedded agents and human-centric computing which he applies to areas such as pervasive computing, ambient intelligence, intelligent buildings and Internet systems. He has authored more than 50 journal papers, conferences and books on these topics. He is a Chartered Engineer holding full corporate membership of the BCS and IEE. Currently, he heads both the Intelligent Inhabited Environments Group and the Brooker Laboratory for Intelligent Embedded Systems at the University of Essex where he lectures on ambient intelligence and pervasive computing.

Graham Clarke has been working in computing since 1970. For some years he has had a strong interest in AI and has participated in a number of research projects, collaborating with Nikola Kasabov on connectionist approaches to classification and Jim Doran on simulating societies. His first degree was in (building) Architecture, and his MSc was in the Applications of Computing. The combination of computing, AI and building architecture which intelligent inhabited environments represent make it a good match to his interests and skills. Since 1995 he has been actively working on the underlying science of embedded agents for intelligent inhabited environments. He has recently received a doctorate in Psychoanalytic Studies which reflects his commitment to the crucial importance of emotion in all human endeavours.

Martin Colley holds a BSc in Computer Science from Queen Mary College, University of London and a PhD in Parallel and Distributed Computing Systems from the University of Essex. He is currently a lecturer in the Department of Computer Science and is responsible for teaching and developing courses on mobile robotics and intelligent machines. He is a member of the intelligent inhabited environments group within the Department of Computer Science. His primary research interests are concerned with the development of parallel and/or distributed control architectures for intelligent environments and machines. In particular, he is interested in the developing control architectures for antonomous vehicles and investigating how these architectures could benefit from interaction with their environment.

Hani Hagras received the BSc and MSc degrees from the Electrical Engineering Department, Alexandria University, Egypt, and the PhD degree in computer science from the University of Essex. He is currently a Senior Lecturer in the Department of Computer Science at the University of Essex. His major research interests are in computational intelligence, notably fuzzy logic, neural networks, genetic algorithms, and evolutionary computation. His research interests also include ambient intelligence, pervasive computing and intelligent buildings. He is also interested in embedded agents, robotics and intelligent machines. He has authored more than 50 papers in international journals, conferences and books. He is a member of the IEEE and a member of the exeuctive team of the IEE's Robotics and Mechatronics Professional Network.

Jeannette Chin received a first class honours degree in Internet Computing from Essex University and a Diploma in Civil Architecture from the Polytechnique Ungku Omar, Malaysia. She has a strong interest in pervasive computing and the notion of anywhere, any time, any person computing. Her primary research interest concerns the exploration of the development of interfaces for pervasive home computing that are sensitive to the human relationships they mediate.

She is a Senior Research Officer in the Department of Computer Science at Essex University where she is researching into a human-centric programming paradigm for lay end users known as task-oriented programming (TOP).

Faiyaz Doctor received the BSc degree in Computer Science and Artificial Intelligence from the School of Computing Science at the University of Birmingham in 1998 and the MSc degree in Computer Science and Artificial Intelligent Agents from the Department of Computer Science at the University of Essex in 2002.

He is currently pursuing a PhD in Ambient Intelligent Systems also at the University of Essex.

His research interests include computational intelligence, fuzzy logic, ambient intelligence, pervasive computing and intelligent buildings.

He is also interested in embedded agents and intelligent machines. He is a student member of the IEE.