Presented at 'Intelligent Environments 2010', Kuala Lumpur, Malaysia, 19-21 July 2010

An Adjustable Autonomy Agent for Intelligent Environments

Matthew Ball, Vic Callaghan, Michael Gardner
School of Computer Science and Electronic Engineering University of Essex, Colchester, UK
mhball@essex.ac.uk

***Abstract*- Autonomy of embedded agents in intelligent environments is highly debated topic; while some believe that agents should have very minimal autonomy and should only act as directly instructed by the user, others consider providing agents with autonomy to be an essential aspect to building intelligent environments. This paper reports on the current progress of our project to enable human users and agents to collaborate in managing intelligent environments as a team. We seek to develop an adjustable-autonomy agent in an effort to explore user acceptance of pervasive computing and the use of autonomous agents therein, as wells as aiming to improve the robustness and reliability of future intelligent environment systems. We present our Adjustable-autonomy Behaviour-Based Agent (ABBA) architecture model and discuss our initial trials with our prototype system, built on a smart home emulator, which demonstrate the plausibility of employing adjustable-autonomy in full-scale intelligent environments and pervasive computing systems.**

Keywords- intelligent environments; human-agent teamwork; pervasive computing; adjustable-autonomy; mixed-initiative interaction

I. INTRODUCTION

People have their own individual needs and preferences, which can differ greatly from each other and may change significantly over time. Hence, many pervasive computing systems, such as intelligent environments, need to be tailored specifically to their user. Obviously, it would not be feasible to have an expert (or indeed a team of experts) develop, tailor and continuously maintain a specific system for each of a large number of users. Instead, two mainstream approaches to this management problem have emerged from recent research that argue intelligent environments should be programmed and managed over time after deployment by embedded-agents, either autonomously by agents or as directly instructed by the end-user. Autonomous-agent driven systems have the advantage that they remove the cognitive load from the user, whilst end-user driven systems have the advantage that, unlike autonomous agents, the system is not required to guess the intentions and needs of the user. We, however, believe that a more ideal system would provide both options for management using human-agent teamwork. In this paper we explore the issues with taking an exclusively autonomous-agent or exclusively end-user driven approach to management, following on from our previous works [1][2]. Section II discusses the exclusive approaches of management and illustrates the problems that one may encounter with them. Section III presents our architecture model for an Adjustable-autonomy Behaviour-Based Agent (ABBA). Section IV explains our prototype system that allows for adjustable autonomy and we described two trials conducted using the prototype that demonstrate the plausibility of employing an adjustable-autonomy agent to manage an intelligent environment. Finally, Section V gives a concluding discussion.

II. THE MANAGEMENT PROBLEM

In this section we describe the two different approaches of end-user and autonomous-agent driven management systems and highlight some issues that may be encountered with them.

When we talk about management, we mean the configuring (forming topographical connections) and programming (customising the functionality) of intelligent environments. Figure 1 shows an abstract view of how a general pervasive intelligent environment is managed. It shows a controller (which could be a program, intelligent agents, etc.), the environment and a user; these are connected in a cycle, which we will refer to as the management cycle. The management cycle starts with the user acting in the environment based on their perceptions and preferences. These preferences are then captured, either implicitly through autonomous sensing or explicitly through end-user programming, by the controller, which drives the effectors and produces actions in the environment based what it has recorded about the user's needs. The cycle is completed by the user perceiving the effects of the controller's actions in the environment and responding in some

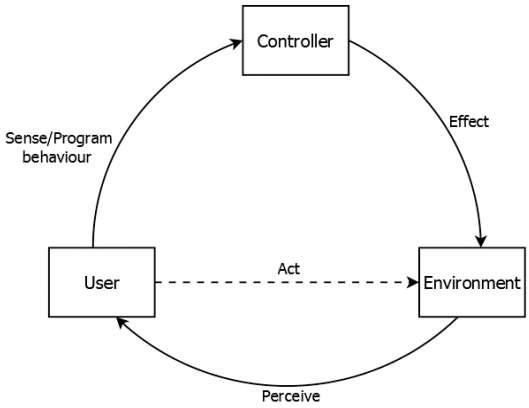


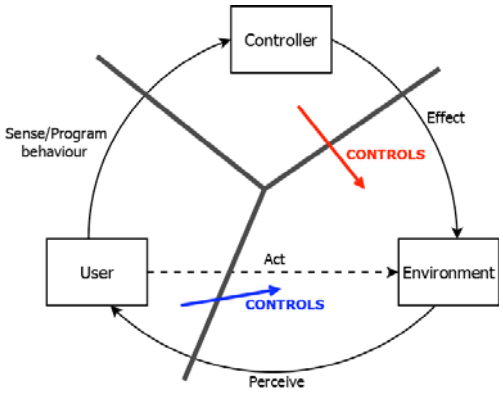
Figure 1. The management cycle

way, which in turn, leads to cumulative loops around the cycle.

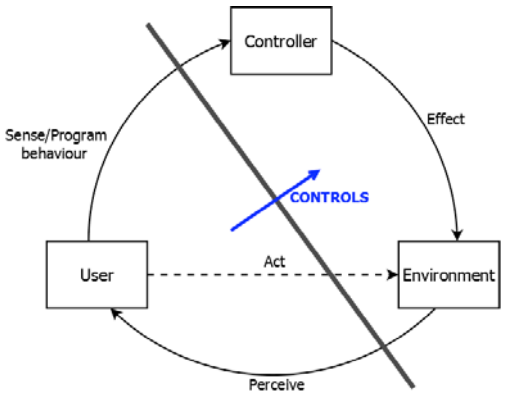
As introduced earlier, in recent research there are two mainstream approaches to IE management. Firstly, an end-user driven approach can be taken. In this approach it is the responsibility of the end-user to program the IE, although the user of the system may not actually have any knowledge of computer programming nor any technical knowledge of the system. To enable the end-user to program the behaviour rules more easily, an end-user driven approach usually adopts a simplified programming mechanism, as in [3][4][5]. The second approach is to make the system autonomous. The system then employs autonomous-agents to program itself by learning from the user's behaviours and interactions with the environment in context with the current environmental [6][7]. End-user driven and autonomous-agent driven approaches can be seen as being at two opposite ends of a scale [2][8]. The Ball Management Contention Diagrams, shown in Figure 2, illustrate the problems of exclusively using one of these approaches of management and how these problems might be overcome by taking a hybrid human-agent teamwork approach.

An end-user driven approach empowers the user, giving them complete control in managing the system while, an autonomous-agent driven system disempowers the user handing complete control over to a collection of agents. In most situations, producing a system that empowers the user might seem the logical choice; however, problems can arise in a fully end-user driven system since the intelligence of the management system is dependent on the creativity, intelligence, willingness and ability of the user. This is depicted in Figure 2(a); the environment and the controller effectively become one entity; the controller becomes little more than an interface to the environment and hence the pervasive intelligent environment relies completely on the user for it to be adaptable and intelligent. Such a complex system cannot rely on all users to be creative, intelligent, willing and able enough to manage the system in all manner of situations; for example users may be too busy or lose confidence in their ability from time to time, or users may well have a physical disability and find it very difficult or even impossible to interact with the computer devices. In these situations, an autonomous system is clearly the superior choice; it greatly reduces the cognitive and sometimes the physical load placed on the user in programming and managing the system. Although, issues may also arise in a fully autonomous management system since there is no direct communication between the user and agent, as depicted in Figure 2(b); the user and the agent (controller) are operating separately although they are effectively working towards the same task – control the environment to suit the user's requirements. At some point the system has to rely on guesswork to assume the user's requirements and from time-to-time an agent will inevitably guesses wrongly, which could be highly annoying to the user or indeed completely unacceptable. Moreover, if the management system is operating in an unknown or restricted environment, it may not be able to get enough information to make a rational decision or take action, which again may lead to the user's displeasure or perhaps a complete failure of the system.

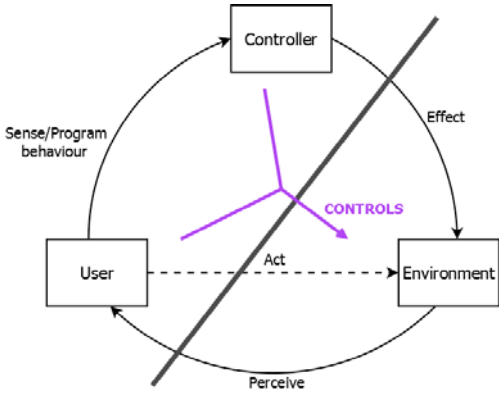
We seek to combine these two distinct approaches and create a hybrid system in which the end-user and autonomous-agents (as the controller) work together as a team, depicted in Figure 2(c). If the user and agent collaborate together, this reduces the chance of guesswork needing to be done and if either the user or agent cannot, for whatever reason, manage the system in the usual way, they can seek help from the other. This creates an overall more robust and reliable management system. Also, the user is no longer forced to either manage and program the environment themselves or be at the complete mercy of autonomous agents, instead they can manage the environment at level at which they feel comfortable doing so.



(a). Separation in an end-user driven system



(b). Separation in an autonomous system



(b). A human-agent teamwork based system

Figure 2. Ball Management Contention Diagrams – An illustration of problems of exclusive management

III. ENABLING HUMAN-AGENT TEAMWORK THROUGH ADJUSTABLE AUTONOMY

In this section we consider how we can enable human-agent teamwork in intelligent environments by employing the concepts of adjustable-autonomy and mix-initiative interaction and we present and describe our architecture model for our Adjustable-autonomy Behaviour-Based Agent (ABBA).

Bradshaw et al. describe adjustable-autonomy as maintaining "the system being governed at a sweet spot between convenience (i.e. being able to delegate every bit of an actor's work to the system) and comfort (i.e. the desire to not delegate to the system what it can't be trusted to perform adequately)" [9]. That is to say, adjustable autonomy allows an agent to 'back-off' and let the user take control of certain tasks that would usually be done autonomously, whenever the user so wishes. A closely related concept to this is mixed-initiative interaction. Mixed-initiative interaction can be defined as two or more parties (for example agents and users) each providing a level of initiative in collaboratively completing a task [9]. Using these two concepts, autonomous systems can be made to 'share' their tasks with human users, so that they are completed as a team. In other areas of AI and robotics, researchers have successfully applied these concepts to enable human-agent teamwork in their systems, for example: Allen and Ferguson's human-machine collaborative planning system [10] allows humans and agents to work together to plan the evacuation of an island and researchers at NASA have developed a mars rover that allows users to take control of specific subsystems at any time whilst all others remain operating autonomously [11].

Figure 3 shows our Adjustable-autonomy Behaviour-Based Agent (ABBA) architecture model that allows for the agent's level of autonomy to be adjusted and enables the user to collaborate in the creation of behaviour rules in intelligent environments. It is inspired by the incremental synchronous learning (ISL) agent developed by Hagras et al. [12]. The ABBA architecture takes the general form of a behaviour-based architecture, as pioneered by Brooks at MIT [13]. In such architectures a number of agent behaviours run in parallel. A controller is employed to coordinate the behaviours or their given outputs into one single output to achieve the desired agent functionality. The ABBA architecture model uses two sets of behaviour rules: one active set and one potential set. Each behaviour rule is assigned with a confidence level. A rule can only have an effect on the environment if it is active and can only be active if it has a high enough confidence level. Rules with a low confidence can only be potential behaviour rules and cannot effect the environment. All behaviour rules are visible to all components of the agent. The behaviour arbiter component regulates the behaviour rules; it reduces the confidence of all rules overtime. If an active rule's confidence level drops below a certain threshold, it will drop down into the potential set and if a potential rule's confidence level drops below a very low threshold (zero for example) then it is deleted. This confidence degradation reduces the chance that the agent's memory will become full. The coordinator regulates/merges the output of all the active behaviours into one single output so that each behaviour rule effects the external environment to an appropriate degree. Additionally, when an active behaviour rule effects the environment the coordinator increases the level of confidence of that rule, where the amount increased depends on the degree the rule is effecting the environment. Thus, the more a rule is used and the more it effects the environment, the less chance it will have of dropping into the potential set and ceasing to be active.

To adjust autonomy of ABBA, the amount of confidence the agent can give to a rule, new or existing, is restricted; both the user and the agent can program behaviour rules in the potential set but depending on the level of autonomy the agent may require the user to add additional confidence to

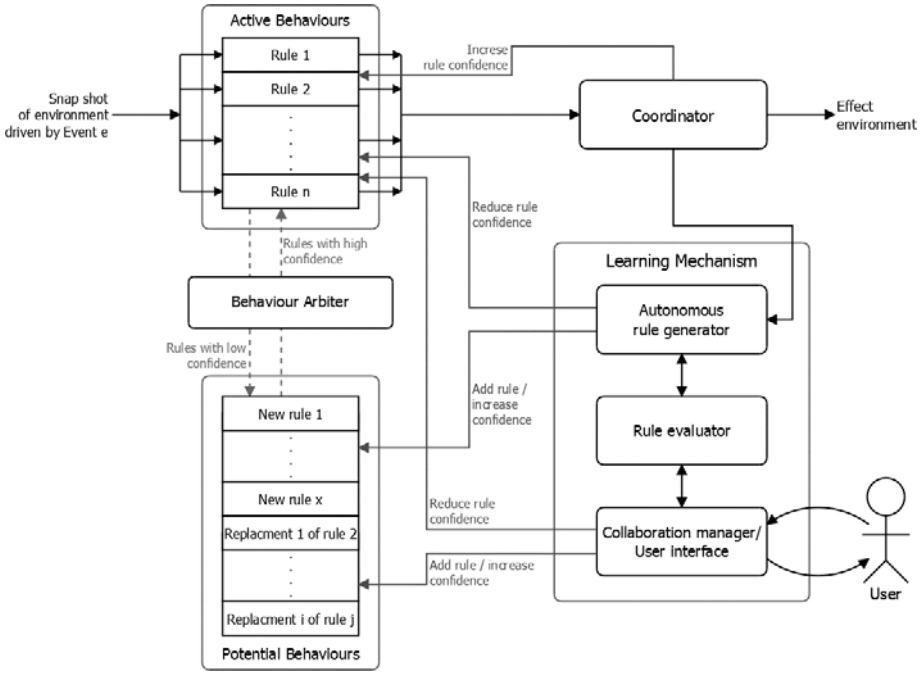


Figure 3. ABBA – Adjustable-autonomy Behaviour-Based Agent

autonomously programmed rules before they can become active. Through this confidence based mechanism we can force the agent to collaborate and hence alter its level of autonomy. By changing the level of autonomy by explicitly restricting the agent's confidence in this way we create a style of task delegation in which the user can choose to adopt or delegate a task to a certain degree. One can imagine a sliding-scale switch (similar to a volume control) that represents the level of autonomy of management: one end of the scale represents fully autonomous-agent driven management and the opposite end represents exclusively end-user driven. The user could use this theoretical sliding scale switch to explicitly say to what degree they wish to contribute to a given task, i.e. to what degree the management should be done autonomously. To further this analogy, one could imagine the pervasive intelligent environment being divided up into sub-systems, for example heating, lighting and security. Behaviour rules could then be categorised into these sub-systems; thus allowing us to create a theoretical mixing desk consisting of many sliding-scale switches to control the level of autonomy throughout the entire system.

Direct collaboration could also be used in the system to 'fine tune' the level of autonomy further. Perhaps the simplest form of collaboration in our system would be for the agent to develop a rule and present it to the user for them to accept or reject it. Here we can say the system is highly autonomous as it requires a lot of initiative from the agent and little from the user. In order to reduce autonomy further, we must increase the level of initiative from the user. This can be done by giving the user the option of altering the rule before accepting it. The same choices could also be offered to the agent if a user has created a rule in a more end-user driven system or if a user suggests an alteration to a rule generated by the agent. Hence, we can create a kind of back-and-forth communication in a way that is quite similar to Allen and Ferguson's collaborative planning system [10]. Each participant will be able to suggest new rules and accept, reject, or alter the others suggested rules. This collaborative mechanism will allow either participant to provide varying levels of initiative to the collaboration; the more a participant makes suggestions for new rules and alterations to the other's suggested rules, the more initiative they provide. Changing the way that the collaborative mechanism is triggered will help to adjust the level of the agent's autonomy. For example, in a more autonomous system, when an agent generated potential rule's confidence level has reached a high enough level, say 75% (i.e. when the agent has seen the user repeat the same action a number of times in the environment), the remaining 25% confidence will have to be gotten from the user before the potential rule can become active and the agent will start a collaboration. Here, since the system requires an input from the user, we can say that the agent is no longer fully autonomous because of an explicit restriction of its confidence in its own generated rules. The agent's level of autonomy is then adjusted further implicitly in the collaboration depending on the level of initiative provided by the user; if the user simply excepts the potential rule we can say the agent operated with a higher level of autonomy than if the user makes an alteration to the agent generated rule. In this way we provide an agent architecture model, in which the level of autonomy is adjusted in two ways: firstly an initial level is explicitly set by the user (task delegation) and the level of autonomy can then be adjusted further implicitly through direct collaboration.

IV. Building A Prototype Agent

In this section we describe our implemented prototype version of ABBA and two trials undertaken that show the plausibility of employing adjustable-autonomy in intelligent environments.

In order to test the plausibility of adjustable-autonomy agents in intelligent environments a prototype version of ABBA has been implemented using two SunSPOT devices interfacing with an mDorm. A SunSPOT (Sun Small Programmable Object Technology) is a small, wireless, battery powered device based on an ARM processor. Using the Squawk Java virtual machine, it allows programmers to create projects that used to require specialized embedded system development skills easily using java. The hardware platform includes an ARM processor, a radio, and a range of built-in sensors as well as the ability to easily interface to external devices [14]. An mDorm is a miniature intelligent environment designed to act as a emulator for the Essex iDorm and iSpace, two real-world intelligent environment test-beds at the University of Essex described in [1]. The mDorm uses two sets of lights, a heater and an extractor fan as actuators and is equipped with temperature and light level sensors. One SunSPOT, the internal SunSPOT, is connected internally to the mDorm; this acts as a controller for the mDorm, monitoring and effecting the current state of the environment directly. The second SunSPOT, the remote SunSPOT, is external to the mDorm but interfaces with it through the internal SunSPOT via a radio communication link. Figure 4 shows a diagram of the prototype set-up and Figure 5 shows an image of the real setup with an mDorm connected to PC to provide text-based output, both the remote and internal SunSPOT devices are circled in green. Our prototype agent is implemented on the remote SunSPOT, external to the mDorm, so that it has the potential to be used with multiple environments (mDorms); this being a major point of further investigation in upcoming stages of our research, as described in [1]. The agent prototype has been implemented as a simplified version of ABBA previously described in section III, with the agent's adjustable-autonomy mechanism being based around a confidence utility. The agent is able to learn behaviour rules autonomously through monitoring of user actions in the environment and can also be programmed with rules directly by a human user using a programming-by-example methodology [15]. A rule in the

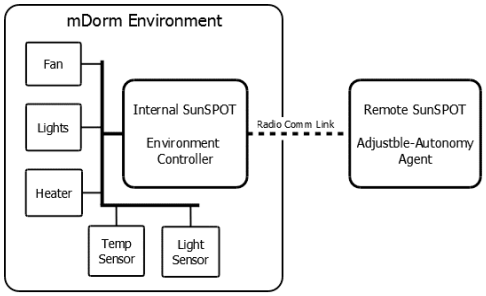


Figure 4. Diagram of the prototype system setup

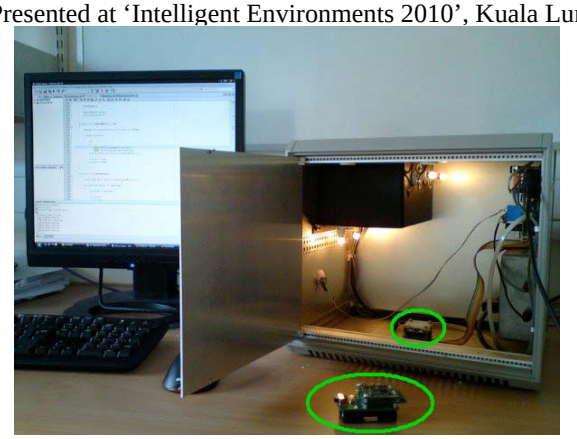


Figure 5. Real-life prototype system setup

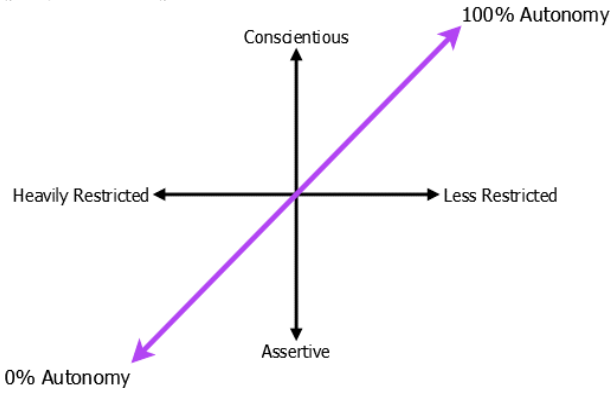


Figure 6. Graph illustrating range of autonomy in the prototype ABBA

ABBA system is a mapping of a state to a set of actions, taking a similar form to:

$$IF\ CurrentState = StateX\ THEN\ DO\ ActionsX \quad (1)$$

Where *StateX* describes a context (a set of environmental conditions) and *ActionsX* describes a set of actions either leant by the agent or programmed by the user that reach a desired environmental state.

The prototype is currently implemented to test the task delegation form of adjusting autonomy – explicitly setting an autonomy level, as described in section III. As this is only a early prototype, the autonomous learning mechanism is an extremely simplistic form of evidential learning; it learns a specific behaviour rule upon observing repeated user actions by increasing the confidence of the rule at each observation of the same action in the same context (state of the environment). In order to adjust the agent's level of autonomy, two of the agent's traits are altered: its confidence restriction and its assertiveness. By placing a boundary on how much confidence an agent can assign to a rule in total, we limit the level of initiative provided by the agent in programming behaviour rules, forcing it to collaborate, and thus the system will require more initiative from the user and become less autonomous. We can make an agent more or less assertive by changing the amount of confidence that an agent can assign to a rule at any one time (each time it observes the given action). A more assertive agent will be able to increase the amount of confidence of a rule more in any single observation of the given action than a less assertive (more conscientious) agent can in any one time. Thus, a more conscientious agent will collect more evidence before programming a rule so that it, one could say, puts more effort and care into what it programs; this will use a higher level of initiative in programming than a more assertive agent and, hence, we can say a more conscientious agent is more autonomous in programming behaviour rules than an assertive agent. Figure 6 illustrates the range of autonomy (depicted by the diagonal line on the graph) along which our prototype agent can be adjusted: as the agent becomes increasingly conscientious and less restricted it becomes more autonomous; conversely, as the confidence restriction placed on the agent becomes heavier and the agent becomes more assertive (basing it decisions on less evidence) it becomes less autonomous.

For simplicity in the prototype, the agent was only implemented to control lighting in the mDorm based on the time of day. The time of day was simulated in the mDorm as being either Morning, Afternoon, Evening or Night; each period lasting ten seconds (pausing when a user wants to change settings or program a rule). Two trials were run on the prototype system to test if the confidence based mechanism can adjust the autonomy of the agent. In both trials the aim was to create the following rule so that it belongs to the active set of behaviours:

$$IF\ CurrentState = \text{“}Night\text{”}\ THEN\ DO\ SetLightsMax \quad (2)$$

In the first trial the agent's autonomy level was set to 100%, meaning it was unrestricted in the total amount of autonomy it could assign to a rule, so it could program an active rule with no initiative from the user, but was also very conscientious, needing to see the same action in the same context repeated a high number of times (six in this small-scale experiment) before it could assign full confidence to the respective rule. In order to complete this trial, the user simply waited until the Night time period and set the level of the lights to its maximum allowed value (240), and then repeated this another five times. The confidence level was then high enough for the rule to become an active behaviour (i.e. have a confidence level greater than 90%). Figure 7 gives a sample of the agent's text-based output; it shows the agent capturing the user's action for the sixth time, the rule becoming active and finally the agent triggering the rule in the next Night time period, instructing the internal SunSPOT to effect the

```
        .
        .
Remote Agent Running
Remote Agent Running
User has changed state:..Night: setLightsTo->240
Incresing confidence of rule:..[0] :- Night:
setLightsTo-> 240
     New confidence level is:..96
     Rule is now active!
Remote Agent Running
        .
        .
   [Omitted Text]
        .
        .
Remote Agent Running
Time is :-Night
Trigger rule:Night: setLightsTo-> 240
        .
```

Figure 7. Example output from first trial

environment.

For the second trial the agent's autonomy level was set to only 50%. Here the agent should be required to take some input from the user to be able to program behaviour rules for the mDorm, as it has a higher confidence restriction (restricted to 50%) and it will also be more assertive than in the previous trial – it will only require to see an action repeated three times before it assigns its maximum allowed confidence. To complete this trial, the user firstly acted in the same manner as he did in the first trial: waiting until the Night period and adjusted the lights to maximum. However, after this was repeated three times, the agent reached its maximum confidence level and prompted the user to accept or reject the rule. The user accepted the rule, which then gave the rule the remaining 50% confidence it needed to become active. The accept-or-reject decision by the user represents a simplistic form of collaboration in the ABBA prototype, as describe in section III. Figure 8 is an example of the text-based output from the second trial; it shows the agent detecting the action for the third time and asking the user to accept or reject the rule as it has assigned the maximum confidence allowed.

In comparison, one can see that in the first trial the ABBA agent is more autonomous than it is in the second. During the first trail the agent relied only on its own initiative to program the active rule and did so by collecting more evidence on which to base its decision, whereas, in the second trial, the agent used a lesser amount of initiative, collected less evidence and hence relied on the additional initiative of the user to make the rule active. We recognise that this implementation of the ABBA prototype is extremely simplistic and very limited in its functionality; however, the success of the trials does demonstrate the plausibility of adding the extra dynamic of adjustable-autonomy into a full-scale intelligent environment and using mixed-initiative interaction to enable human-agent teamwork.

V. CONCLUSION

The vast majority of previously researched intelligent environment management systems have taken either an exclusively autonomous-agent or end-user driven approach. Although these research efforts have been fruitful, taking such an exclusive approach will undoubtedly cause problems for some users and in certain situations as pervasive computing technology develops further. Hence, we argue that future management systems of intelligent environments must allow for human-agent teamwork and adjustable-autonomy. This will be especially important for users with very specific needs that can change unexpectedly overtime, such as those with physical disabilities and deteriorating medical conditions; the extra dynamic of adjustable-autonomy will enable future systems to be highly customisable to their users and to be more robust and reliable in certain situations. In this paper we have presented ABBA – an Adjustable-autonomy Behaviour-Based Agent architecture model and discussed our initial prototype system, built on a smart home emulator, in which the agent's level of autonomy can be adjusted by tuning two different variables: assertiveness of the agent and a maximum confidence restriction. The successful trials of the prototype demonstrate the plausibility a fully-functional adjustable-autonomy intelligent environment.

```
          .
          .
    [Omitted Text]
          .
          .
Remote Agent Running
User has changed state:..Night: setLightsTo->240
Incresing confidence of rule:..[0] :- Night:
setLightsTo-> 240
Autonomous agent has assigned maximum confidence...
Confidence level is now:..50
Do you wish to accept or reject the rule?
Press switch 1 for accept or switch 2 for reject....
Switch 1 :- User accepts rule
     New confidence level is:..100
     Rule is now active!
Remote Agent Running
          .
```

Figure 8. Example output from second trial

REFERENCES

[1] Ball, M., Callaghan, V., Gardner, M., Trossen, D., "Achieving Human-Agent Teamwork In eHealth Based Pervasive Intelligent Environments", in *Proceedings of 4th International Conference on Pervasive Computing Technologies for Healthcare 2010*, Germany, IEEE Xplore (2010).

[2] Ball, M., Callaghan, V., Gardner, M., Trossen, D., "Exploring Adjustable Autonomy and Addressing User Concerns in Intelligent Environments", in *Proceedings of the 5th International Conference on Intelligent Environments 2009*, Spain, IOS Press (2009).

[3] Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.P., Koleva, B., Rodden, T., Hansson, P., "'Playing with the Bits', User-Configuration of Ubiquitous Domestic Environments", Proceedings of UbiComp 2003, Springer-Verlag (2003), 256-263.

[4] Gajos, K., Fox, H., Shrobe, H., "End user empowerment in human centered pervasive computing", in *Proceedings of Pervasive 2002*, (2002), 1-7.

[5] Chin, J., Callaghan, V., Clarke, G., "An End User Tool for Customising Personal Spaces in Ubiquitous Computing Environments", in *Lecture Notes in Computer Science: Ubiquitous Intelligence and Computing*, Springer-Verlag (2006), 1080-1089.

[6] Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., Duman, H., "Creating an Ambient-Intelligence Environment Using Embedded Agents", in *IEEE Intelligent Systems, Vol.19, No.6*, (2004), 12-20.

[7] Mozer, M.C., "Lessons from an adaptive home", in *Smart Environments: Technology, Protocals and Applications*, Wiley (2005), 273-298.

[8] Callaghan, V., Clarke, G.S., and Chin, S.J.Y., "Some Socio-Technical Aspects Of Intelligent Buildings and Pervasive Computing Research", in *Intelligent Buildings International Journal, Earthscan, 1:1*, (2008).

[9] Bradshaw, J.M., Feltovich P.J., Jung, H., Kulkarni, S., Taysom, W., Uszok A., "Dimensions of adjustable autonomy and mixed-initiative interaction", in *Agents and Computational Autonomy: Potential, Risks, and Solutions, (M. Nickles, M. Rovatos, and G. Weiss, Ed.)*, Springer-Verlag (2004), 17–39.

[10] Allen, J., Ferguson, G., "Human-Machine Collaborative Planning", in *Proceedings of the NASA Planning and Scheduling Workshop*, (2002)

[11] Dorais, G. A., Bonasso, R. P., Kortenkamp, D., Pell, B., and Schreckenghost, D., "Adjustable autonomy for human-centered autonomous systems on Mars", In *Proc. Of the First International Conference of the Mars Society*, (1998).

[12] Hagras, H., Colley, M., Callaghan, V., Clark, G., Duman, H. and Holmes, A., "A fuzzy incremental synchronous learning technique for embedded-agents learning and control in intelligent inhabited environments," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, HI, 2002, pp. 139–145.

[13] Brooks, R., "Intelligence Without Representation", in *Artificial Intelligence*, (1991) 47:139-159.

[14] Sun Microsystems, “Frequently Asked Questions”, in *SunSPOT World*, accessed onine March 2010, at www.sunspotworld.com/docs/general-faq.html

[15] Lieberman, H., “Your Wish Is My Command: Programming By Exampl”, Morgan Kaufmann, San Fransico (2001).