In the International Transactions on Systems Science and Applications (special issue on "Agent based System Challenges for Ubiquitous and Pervasive Computing"), Vol 4, No. 1, 2008

Facilitating the Ambient Intelligent Vision: A Theorem, Representation and Solution for Instability in Rule-Based Multi-Agent Systems

Victor Zamudio and Vic Callaghan

University of Essex, Department of Computer Science
Wivenhoe Park, Colchester CO4 3SQ
United Kingdom
Email: {vmzamu; vic}@essex.ac.uk
http://iieg.essex.ac.uk

Abstract: Multi-agent systems underpin the vision for ambient intelligence. However, developing multi-agent systems is a complex and challenging process. For example, pervasive computing has been found susceptible to instability, due to unwanted behaviour arising from unplanned interaction between rule based agents. This instability is impossible to predict, as it depends on the rules of interaction, the initial state of the system, the user interaction, and in the time delay of the system (due to network traffic, different speed of processing, etc). In this paper we present a theoretical framework, an Interaction Network (IN), together with a communication locking strategy that we call INPRES (Instability Prevention System) that can be used to identify and eliminate this problem. In addition we describe a Multi-Dimensional Model (MDM) to represent the agents and the state of each agent over time. A theorem showing the role of delays in an unstable system is presented. We present experimental results based on simulations and a physical emulation that demonstrate the effectiveness of these methods.

Keywords: agent challenges, pervasive computing, instability, periodic behaviour, multi-agents.

1. Introduction

Multi-agent systems underpin the vision for ambient intelligence. At the heart of this vision is the interconnection of vast numbers of networked devices such as lights, heaters, TVs, telephones, etc., each programmed according to a certain rules based on the state of the world, including other devices These interconnections enable the system to be programmed with interdependent actions in a simple way, whether it be manual or automatic [7], [12], [19].

Pervasive computing is related to other fields, such as distributed systems (eg personal computers connected via a local area network) and mobile computing (a distributed system with mobile clients), but goes much further, as it could involve more complex characteristics: effective use of smart spaces, invisibility, localized scalability and masking uneven conditioning [32]. Autonomous agent managed smart spaces are able to adjust the environment based on the user's preferences, in a proactive fashion, with minimal user intervention.

The overlap between pervasive (or ubiquitous) computing and intelligent agents has spawned the emerging area of Ambient Intelligence (AmI), a new multidisciplinary paradigm, which includes architecture, electronics, robotics, machine learning, etc [30] which has given rise to numerous new challenges.
Pervasive computing has opened the opportunity to extend the use of the internet and other emerging technologies to control everyday environments. For example it can assist elderly people, monitoring their activities and provide them with reminders, reports and control of devices [20] and minimise home energy consumption thereby helping to counter the problem of climate change [8].

In rule-based multi-agent systems, it is possible to have unwanted outcomes, due to the rules of behaviour of the agents. The set of agents could be defined in such way that the agents cyclically repeat their state, showing an oscillatory behaviour. In that case, we say that the system is instable (see section 3)

The problem of instability in intelligent environments is very challenging, not only due to the complexity of the rules of the interconnected devices and non-deterministic user interaction, but also because of temporal delays (network latency, speed of processing, etc) which, for example, is exasperated by the use of nomadic, devices. These temporal delays could contribute to unstable cyclic behaviour. We have seen this phenomenon in our own systems (EU eGadgets Project [5]) and it is being observed increasingly in pervasive computing system as the architectures move from centralized to distributed control [17].

Other domains, such as complex and dynamic systems, have addressed the dynamics of massively interconnected systems. In that work it has been shown that it is not possible to determine what they term *attractors* (an infinite loop in the state space) and the *basin of attraction* (the set of configurations which converge toward an attractor), for an arbitrary Boolean network [36]. This problem has a direct relationship to interconnection topologies we are addressing in our pervasive computing environment. Additionally, the arbitrary rules given by the user (defining the topology of the system) and the perturbations to the system (when the user interacts with the environment) add significant complexity to the dynamics or the system. However, it is possible to detect and prevent cyclic instability and we have developed and tested such a strategy, based on algorithms for finding loops in an Interaction Network and locking devices with the least functional impact on the performance of the system.

1.1 Related Work

Instability can be present in a number of different domains, from home automation [3], to context-aware systems where the user is part of the control loop; this is the case of the Anti-locking Braking System ABS, which is a context-aware system where the driver is playing a fundamental part in the process of preventing an accident [11]. If the driver uses the traditional method of pumping the brakes with a car equipped with ABS, a conflict can arise causing the breaking distance to be increase, due to the problem of failing to reach a stable state. In home automation cyclic instability was observed in the EU Project, CUSTODIAN (Conceptualisation for User involvement in Specification and Tools Offering the Delivery of system Integration Around home Networks) which used boolean functions (logical condition that must be TRUE for the device to be activated). In this system a single module was responsible for the propagation of any changes in the devices, with every smart device changing its status as appropriate. They observed that on occasions the network didn't stabilize requiring them to terminate the process so that the network could be debugged [26].

Software agents, such as those employed in ecommerce. may be involved in loops with other agents. A simple, but well known example occurs in email lists, where users have configured auto-replays that answer each other [27].

In telephony, there is a well-reported issue referred to as 'the feature interaction problem [3], which occurs when a customer or customers have several active features (such as call-forwarding, extension dialling, call-waiting, etc) which together interfere with or otherwise influence each other's functionalities or behaviours [4]. For example consider two features: Calling Number Delivery (CND), which delivers the calling party's directory number to the called party and Unlisted Number (UN) that prevents a subscriber's user number from being released. Suppose a subscriber A with UN places a call to another subscriber B with CND. If the network allows A's number to be delivered to B, then A loses privacy; if it does not, then B gets no information. Either way, one feature does not work desirably [9]. In home automation this manifests itself in various ways. For example a burglar alarm might be activated when fire breaks out which in-turn could close all the doors whilst the fire alarm service would try opens all doors. Thus both services might try to control the doors in conflicting ways, interfering with one another, causing unexpected behaviour. In a similar way, other services, independently developed, could be available (security, entertainment, climate control, etc) which could cause conflicts and undesirable outcomes. Another example concerns concurrent heating and air conditioning services where, on reaching a certain temperature, the air conditioning is turned on, causing the temperature to drop below a certain level and triggering the heating service resulting in periodic behaviour [24], [38], [37].

In computer hardware and system design there are well known instability problems, such as race hazards and metastability, which can lead the system to display unwanted behaviour. Race hazards are caused by logic signals taking differing paths, with differing delays, through digital circuits which displace them in time causing unexpected combinations of states with consequent generation of spurious momentary logic states. Metastabilty is caused by violation of set-and hold times in asynchronous systems which can have various outcomes ranging from spurious states, undefined states or oscillations [25] [34]. This work has shown that whilst, in general, there are solutions for synchronous systems, instability intrinsic to asynchronous design and cannot, in general, be entirely eliminated [34], of which the pervasive computing problems we are addressing relate to.

In distributed computing systems there is a similar problem, related to the assignment of resources to different users, known as deadlock. A deadlock occurs when two or more processes are waiting indefinitely for an event that can only be completed by one of the waiting processes, or based on circularity of definitions [21]. When using a WAITFOR or bipartite graph this problem can be found when a directed cycle is detected and a transaction could be selected to break the cycle [14]. Deadlock can be seen as the opposite problem to the one we are addressing; in the case of a deadlock every device in the loop is static, but in our problem every device or agent in the loop is working indefinitely.

Planning deadlock problems occur in multirobot systems, when a robot enters into a waiting cycle, where it should wait for a response or for calculations performed by a set of robots that it is part of. This can be detected if in the planning dependency graph, a cycle is found, anticipating and avoiding a deadlock during the execution phase where backtracks are not always possible. A simple example of this is when two robots have the goal to move to the initial condition of the other one [29]

Another notable example occurs in amorphous computing and small worlds [35][1]. Both are examples of nano scale computing systems which are massively connected asynchronous systems in which their topologies have a probability of connection between ordered and random.

Social networks such as the interaction between people and organisations are another area where inter-agent instability can occur. For example, in the world of business instability has been observed between share-traders or even between global economies as people seek to monitor each others behaviour with the intention of modify their own behaviour to give them an advantage [2] Likewise fashion can be seen as an example of this phenomena with fashions oscillating over long periods of time. In a wider sense all living entities can be seen as systems of coordinating agents varying from bacteria to people [35]. Thus, in principle, the issues address by this work (Interaction Networks) is applicable to such systems, but more research would need to be done before the viability of this possibility could be established.

2. Task Representation in Pervasive Computing Space

In order to visualize and reason about the task being performed by the devices in an intelligent environment, we have developed a Multi-Dimensional Model (MDM). With this model we are able to represent, in a graphical way, the binary state of the devices, and their evolution over time thereby providing valuable information about the dynamics of the system.

As it will be seen, this model includes complex devices (which can perform more than one task), showing the temporal evolution of the system. This framework takes into account the mobility of the user and the possibility to provide the same set of coordinated services in a new environment, allowing the user to be presented with different views of the task spaces [40].

2.1 A Multi-Dimensional Model (MDM) of Pervasive Computing Space

We have developed a model of Pervasive Computing space taking into account the following characteristics:

a) Simple devices vs. complex devices. A simple device can only perform one type of task, and can only perform one task at a time. Complex devices can perform several kinds of tasks at a time.
b) Temporal tasks vs. non-temporal tasks. A temporal task depends on time (eg are valid for a specific period). Non-temporal tasks do not depend on time.
c) Coupled tasks vs. uncoupled tasks. Coupled tasks have a mutual interdependency (eg are logically linked). Uncoupled task have no mutual dependency).
d) Static vs. dynamic environment. In a static environment, apart from system failure, devices do not move in time or space. In a dynamic environment devices come and go from the network.

In the next section we will formalize the problem, defining an allocation, a community, and an equivalent community. We will then extend these representations to include time.

2.2 Formalising the MDM Model – Allocations and Communities

An ***allocation*** is a duple (d,T) where d is a device and T is a not empty set of k tasks, *i.e.* $T=\{t_1,t_2,t_3,\ldots,t_k\}$, with $k\geq 1$. If $k=1$ we have a simple device, that is able to handle only one kind of task. This is the case of an audio-speaker, or a microphone. If $k>1$ then d is a complex device, which is composed by other sub-devices, therefore d can handle more than one task. This could be the case for a TV, composed by a device that can handle two different kinds of signals: audio and video.

When the user configures a new set of virtual appliances, he defines a new ***community***. A ***community***, denoted by C, is a finite non-empty collection of n allocations, *i.e.*

$$C=\{(d_1,T_1),(d_2,T_2),(d_3,T_3),\ldots,(d_n,T_n)\} \tag{1}$$

If the user goes to a new environment, the agent should create an ***equivalent community*** C_{eq}. In order to create this equivalent community, for each allocation $(d,T)\in C$ the agent should find an equivalent allocation (d_{eq},T_{eq}) in the new environment. As we mentioned before, we have two cases: $k=1$ and $k>1$.

i) If $k=1$ then d is a simple device and $T=\{t_1\}$. The agent should find a new allocation $(d_{eq},\{t_1\})$ such that the device d_{eq} is able to perform the only task t_1.

ii) If $k>1$ then d is a complex device, and $T=\{t_1,t_2,t_3,\ldots,t_k\}$. The agent should find, in the worst case, k allocations $(d_{eq}^1,\{t_1\}),\ldots,(d_{eq}^k,\{t_k\})$, where every device d_{eq}^i is able to perform the task t_i, with $1\leq i\leq k$.

2.3 Formalising the MDM Model – Temporal Communities

We can extend this framework in order to include time. A ***temporal allocation*** is a tuple (d,T,t_i,t_f) where d is a simple device, T is a (simple) task, t_i is the initial time and t_f is the final time. In other words, the device d will be performing the task T during t_f-t_i units of time, beginning on the instant t_i.

So, a ***temporal community,*** denoted by C_t is a non-empty set of temporal allocations:

$$C_t=\bigcup_{j=1}^{k}\{(d_j,T_j,t_{ji},t_{jf})\} \tag{2}$$

2.4 Applying the MDM Model to practical environments

In Fig. 1 we can see the graphical representation of a system with the following devices: chair-sensor, bed-sensor, window blinds, bed-light, desk-light, heater, word and mp3 player. All of them can be on a binary state on/off. The time runs from 0 to 100.

As we mentioned before, some devices (with their tasks) could be coupled, in the sense that there is a logical link or causal dependency between them.

The MDM model enables us view the status of each device, and its evolution over time. If the MDM is representing a rule-based system, it is possible to visualise different configurations depending on the rules given by the user. Under some circumstances, it is possible to visualise the formation of clusters of devices, due to the logical interdependencies between them. This model allows a simplified visualisation and understanding of the task spaces; for example if the MDM is projected to the Agent-State plane, states not visited (and visited) by the agents can be easily identified. With this it is possible to process and reason about intuitive information such as device-task, device-time, or even just single information such as a device or task. This representation simplifies the difficulties related to dealing with the complexity of the devices, temporality of tasks and dynamics of the environment, and opens up a way to reason about multiple tasks and their interaction.

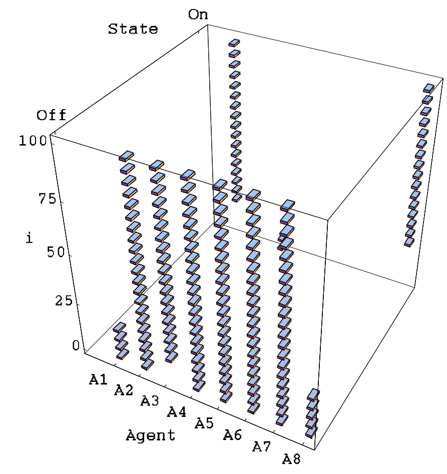


Fig. 1.MDM showing the evolution of a system with 8 binary devices.

The **MDM** Model will be applied in section 3.

3. Interaction Networks

In recent years, the importance of modelling relationships and, in particular, relationships of dependencies in pervasive computing has grown. A significant reason for this growth is that, without this information, it has been shown that decisions made by context-aware applications can be inappropriate or even lead the system to instability [22],[31]. To capture relationships of functional dependencies between the rules of behaviour of the agents, based on a graphical representation, we developed a methodology we refer to as an Interaction Network which is based on directed graph theory.

In this a *directed graph* G consists of a finite set V of vertices or nodes, and a binary relation E on V. The graph G is denoted as (V,E). The relation is called the adjacency relation. If w is relative of v (ie, $(v,w)\in E$) then w is adjacent to v [18]. An agent A is an autonomous device with a binary state $s \in \{0,1\}$, where 0 means that the agent is *off*, and 1 means that the agent is *on*. If we have n autonomous devices agents $A_1, A_2, \dots A_n$ the state of the system is $S = (s_1 s_2 \dots s_n)$. Each agent A_i has two rules: i) if ϕ_i then $s_i = 1$ ii) If ψ_i then $s_i = 0$ where ϕ and ψ are boolean functions that depend on the states of the agents.

An *Interaction Network (IN)* is a directed graph (V,E) in which the vertex $v \in V$ is a pervasive autonomous agent A and $(v_i, v_j) \in E$ if the Boolean functions ϕ_j or ψ_j of the pervasive autonomous agent A_j depends on the binary state s_i of the agent A_i. Let $U \subseteq S$ be a subset of S. Because of the dynamics of the system, the system will produce a sequence of states $U_1, U_2, \dots U_p$. If this sequence of states is periodic then the subsystem U is said to be *periodic*.

The *functionality of a node* is defined as the number of descendants in the Interaction Network. This characteristic of a node is very important, as it shows the impact of a device in the system, in terms of the number of devices whose rules could be triggered. Fig. 4 provides an example of an Interaction Network, showing the dependencies of 5 devices or services: Sofa Sensor, Light Sensor, MP3 Player, Light, and Word.

3.1 Adding Delays to the system: A Theorem involving 2 agents

It is possible to make a refinement in order to include delays in the communication between the agents in the previous model. For this, for each edge $(v_i, v_j) \in E$ we are going to define a delay $w_{ij} \in Z^+$ which means that if the state of agent A_i is updated, the agent A_j is going to evaluate their rules after w_{ij} units of time.

As we have mentioned before, delays in the communication between agents, because of network related delays (eg different paths, network component processing, etc.) could cause some instabilities. We have investigated this with a small system involving 2 agents, with very simple rules of interaction [41]. The rationale is that this model is easier to understand and reason on, but that the findings are scalable to more realistic topologies. This is summarized in the following theorem:

Theorem 1. Let A_1 and A_2 be two agents as defined before, with rules defined by the boolean functions $\phi_1 = s_2$, $\psi_1 = \overline{s_2}$, $\phi_2 = s_1$, $\psi_2 = \overline{s_1}$, and delays $w_{12} = n$ and $w_{21} = m$, with $n, m \geq 2$ (see Fig. 2). Originally the state of the system is $S = (0,0)$. If at $t = 0$ the state is $S = (1,0)$ then the system is periodic, with period $T = n + m$.

Proof. At $t = 0$ the system is in state $S = (1,0)$, and because of the delays, A_1 and A_2 should process, according to their rules, the first element of the string $b_1 = 0_1 0_2 \ldots 0_m$ and $b_2 = 0_1 0_2 \ldots 0_{n-1} 1_n$ respectively. With this, A_2 will be processing the state $s_1 = 1$ after n units of time. At $t = 1$ the system will be in state $S = (0,0)$, and $s_1 = 0$ and $s_2 = 0$ will be added at the ends of the strings b_2 and b_1 respectively (new information should be processed after a delay, according to each agent), ie. $b_1 = 0_2 \ldots 0_m 0_{m+1}$ and $b_2 = 0_2 \ldots 0_{n-1} 1_n 0_{n+1}$. Let's suppose, without any lose of generality that $n < m$. Because the next states to be processed by the agents are all 0's, all the following states will be $S = (0,0)$. At $t = n - 1$ A_1 will be processing the first element of $b_1 = 0_n 0_{n+1} \ldots 0_m \ldots 0_{m+n-1}$ and A_2 the first element of $b_2 = 1_n 0_{n+1} \ldots 0_m \ldots 0_{2n-1}$ and then at $t = n$ the system will be in state $S = (0,1)$, with $b_1 = 0_{n+1} \ldots 0_m \ldots 0_{m+n-1} 1_{m+n}$ and $b_2 = 0_{n+1} \ldots 0_m \ldots 0_{2n-1} 0_{2n}$. Because the next states to be processed by the agents are all 0's, all the following states will be $S = (0,0)$. At $t = m + n - 1$ the state of the system will be $S = (0,0)$ with $b_1 = 1_{m+n} 0_{m+n+1} \ldots 0_{2m+n-1}$ and $b_2 = 0_{m+n} 0_{m+n+1} \ldots 0_{m+2n-1}$. Therefore, at $t = m + n$ the system will be in state $S = (1,0)$ and $b_1 = 0_{m+n+1} \ldots 0_{2m+n-1} 0_{2m+n}$ and $b_2 = 0_{m+n+1} \ldots 0_{m+2n-1} 1_{m+2n}$ which is the same situation as at $t = 0$. All the process will continue exactly in the same way, and therefore the system is periodic.

Figure 2. illustrates the evolution of the system. It can be seen that if n or m m are 1, the evolution of the system would lose a chain of states $(0,0)$. If we have $n = m = 1$, the system will be oscillating between the states $(1,0)$ and $(0,1)$.

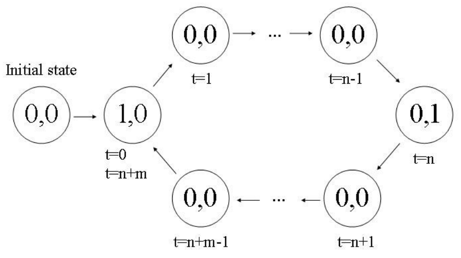


Fig. 2. Evolution of the periodic system. In each node, the first element is the state of the agent A_1 and the second element is the state of agent A_2. The systems has a period of $T = n + m.$

4. Instability Prevention System (INPRES) and Intelligent Locking

In pervasive environments, rule-based devices could interact according to the rules programmed independently by several users. These complex rules, together with the state of the system and the temporal delays could lead the system into unwanted instable states (oscillations). As we have commented previously, it is not possible, in general, to predict if a set of rules will produce such instabilities, as in any dynamic system, the behaviour of the system will depend not only on the rules, but also on the initial conditions. Besides that, the user could interact with the environment, generating perturbations to the system. However, by finding loops in the associated Interaction Network it is possible to identify where potential instability and cyclic oscillations reside so that action can be taken to avoid unwanted behaviour. Our strategy[1] to prevent this unwanted behaviour is based on the algorithm shown in Fig. 3.

```
Cycles C = findCycles(Graph g) ;
for each cycle c in C :
   find  node  n  in  c  which  minimizes
functionality(g, n, c);
   lock n;
od;

Cycles findCycles(Graph g):
   Construct an empty vector of cycles C;
   Construct an empty vector of nodes N;
   for each node n in g do:
         N = descendants(g,n);
         if N.lastElement() == n then:
                 C.add(N);
         fi;
   od;
   return C.removeRepeteadCycles();
```

[1] Patent No: GB 0624827.2.

```
int functionality(Graph g, node n, cycle c):
   Construct a graph' deleting the nodes in c and adding the node n, g'= g - nodes in c + n;
  return numDescendants(g',n);
```

Fig. 3. Intelligent Locking (High-level algorithm). The function findCycles() returns the cycles in the graph, and the function functionality() computes the number of descendents of a node.

This algorithm was programmed in Java. The function called *functionality( ),* calculates the impact of a node (measured as the number of descendants) on the entire network; in each loop, the node which minimises this function is the one that will be locked. The function called *findCycles( )* maintains a list of descendants for each node n of the graph; if the same node n is detected as part of the descendants, a loops has been found. This algorithm can be extended to address the case of more complex graphs. At the moment the algorithm has a policy for automatically locking the nodes, and we are working on an approach to capture the user's preferences in the locking mechanism.

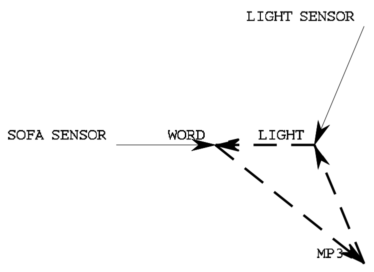


Fig. 4. Interaction Network showing the dependencies between five devices. A loop is shown in dashed lines.

5. Experimental Results

The proposed solution was evaluated in two ways, first with computer simulations and secondly using a real UPnP (Universal Plug and Play) implementation based around the Siemens Java SDK for UPnP technologies [33].

5.1 Simulations

The simulator was programmed using Mathematica ™ **5.1** [39], a programming language with powerful tools for quick and sound implementation. In particular, it includes the package Combinatorica, supporting graph theory, graphics, and combinatorics [28]. The simulation had the advantage over the real implementation (see next section) that it was able to mimic larger numbers of devices and support a more flexible experimental structure (eg arbitrary devices and rules could quickly be created).

Using Mathematica™, a number of parameters can be controlled, for example the number of agents involved, the number of iterations, the probability of perturbations, the probability of interconnection between two agents. In order to test the general approach, we generated random topologies of differing densities (controlling the probability of connection between agents). It is well known that the gates AND and OR (in conjunction with the negation) are able to reproduce any Boolean function. Using this principle, we assigned a random (and fixed) number of boolean function to each device, as a rule of behaviour; thus the rules assigned to each agent could be represented as a binary string, where a 0 and 1 mean an OR and AND gate respectively. As mentioned before, besides the rules of interaction, one of the key factors involved is the initial state of the system (in this case, we always begin with a random initial state) which is then perturbed by user actions.

One of the important parts of the algorithm, besides finding loops, is the process of choosing which agent (or node) to lock. For each loop we would normally need to calculate the functionality of each node. However, in our experiment, the functionality of all the members of a loop is the same and so we excluded the descendant members of the loop from the calculations. We tested our approach successfully with different and randomly produced topologies and rules of interaction, together with random perturbations.

In Fig. 5 we show an Interaction Network with 5 agents, and no cycles. The rules of interaction are coded as {0,1,1,0,1}, where 0 represents an OR function, and 1 represents an AND function. No cycles or loops are present in the IN, and as we can see from Fig. 6, the system is stable, and locking is not needed.

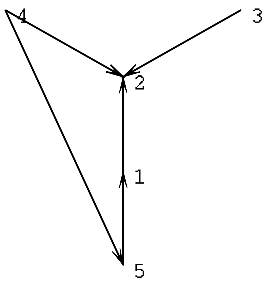


Fig. 5. Interaction Network without any cycle.

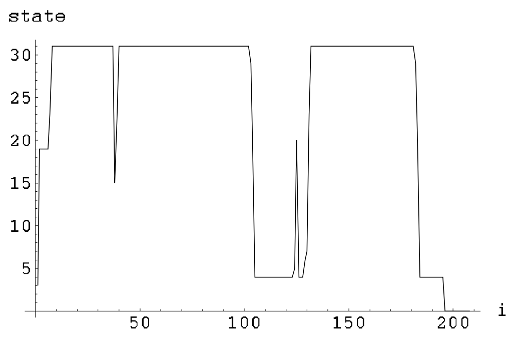


Fig. 6. Evolution of the system with 5 agents and no cycles, showing a stable system.

The MDM of the system can be seen in Fig.7, illustrating the state of each agent from iteration 0 to 20. Initially, some agents are turned off, but after some iterations all the agents are on, which is consistent with the representation on Fig. 6.

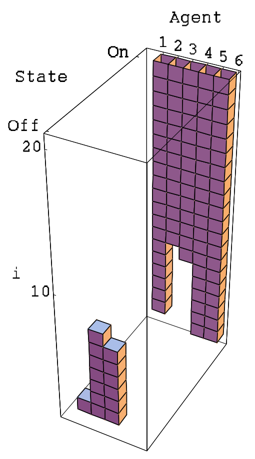


Fig. 7. MDM of the system with 5 agents showing an stable evolution.

Fig. 8 illustrates a system with 7 nodes. This topology has one loop {6,4,6}. The rules of interaction are coded as {0,0,0,1,0,1,0}.

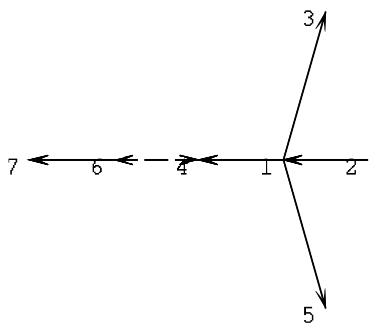


Fig. 8. Interaction Network with only one cycle {6,4,6}. Detail of the cycle in dashed lines.

Node 6 has 1 descendant, and node 4 has no descendants (as we do not include members of the loop). Node 4 has the minimum functionality and, as a result, the locking vector is {1,1,1,0,1,1,1}. Fig. 9 shows the response of the system without any locking, showing cyclic instability; this instability can be removed effectively using the locking mechanism (see Fig. 10).

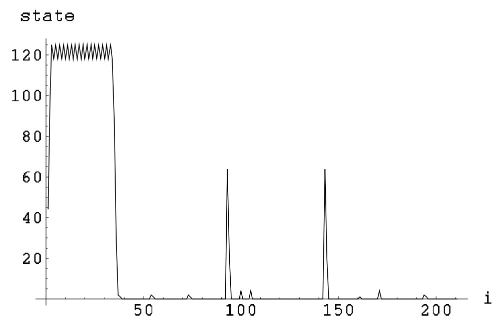


Fig. 9. Evolution of the system with 7 agents and one cycle {6,4,6}.

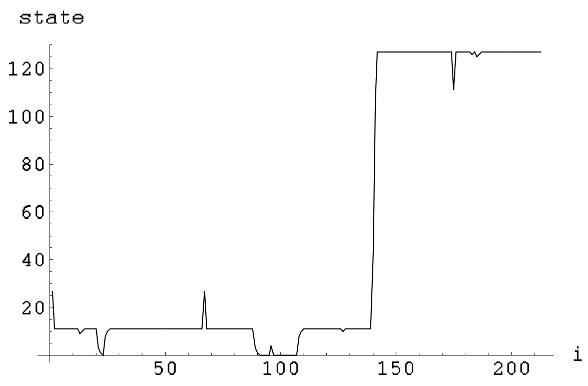


Fig. 10. Evolution of the system with 7 agents, one isolated cycle {6,4,6}, where the node 4 has been locked, and the oscillations have been removed.

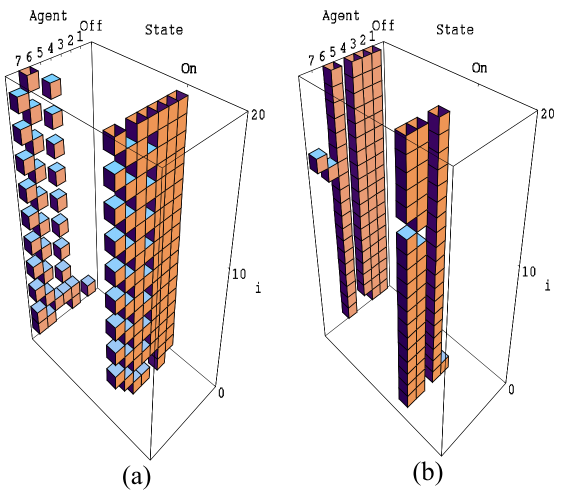


Fig. 11. Evolution of the system with 7 agents between iteration 0 and 20. In (a) there is a periodic pattern. This is consistent with the perturbation shown in Fig. 9. In (b) the instabilities have been removed. A perturbation can be seen after 10, which is consistent with Fig. 10.

As it can be seen, the MDM permits the visualisation of unstable patterns involving a subset of the devices present in the environment.

In Fig 12 we have a system with 10 nodes and two cycles {{8,5,6,7,8},{4,2,3,5,4}}, that share the node 5. The rules of interaction are coded as {0,0,0,0,1,1,1,1,1,0}. Before the locking mechanism was activated, the system displayed instabilities (see Fig. 13). For our graphical representation we use the decimal equivalence of the binary representation of the global state of the system. The list of parent-descendants for the loops are {{4,1},{2,1},{3,2},{5,4}} and {{8,0},{5,6},{6,1},{7,0}}. In the first loop the two nodes 4 and 2 both have the minimum number of descendants (1) and 4 is taken. In the second case 8 and 7 minimizes the functionality function, and 8 is taken. With these results the locking vector for the system is {1,1,1,0,1,1,1,0,1,1}. We ran the simulation several times, and our strategy removed the oscillations, as can be seen in Fig. 14.

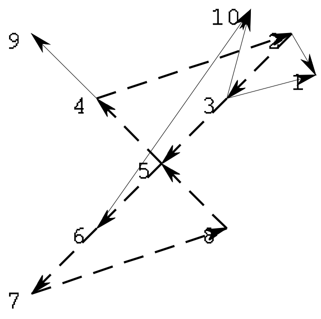


Fig. 12. Interaction Network with 10 nodes and two coupled cycles. The dashed lines depict two loops, sharing node 5.

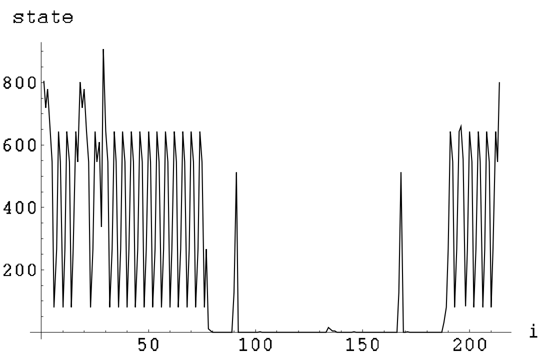


Fig. 13. Behaviour of the system with 10 agents. Oscillation can be seen, together with some perturbations. In this case the locking mechanism had not been applied.

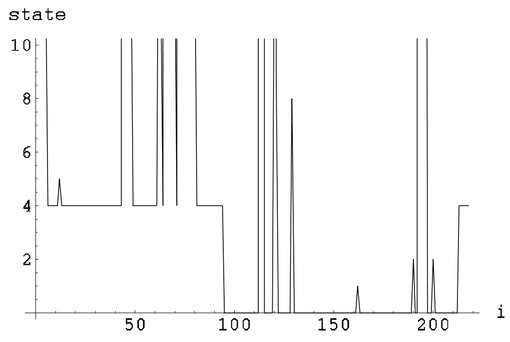


Fig. 14. Behaviour of the system with 10 agents. The instabilities have been removed by the locking mechanism.

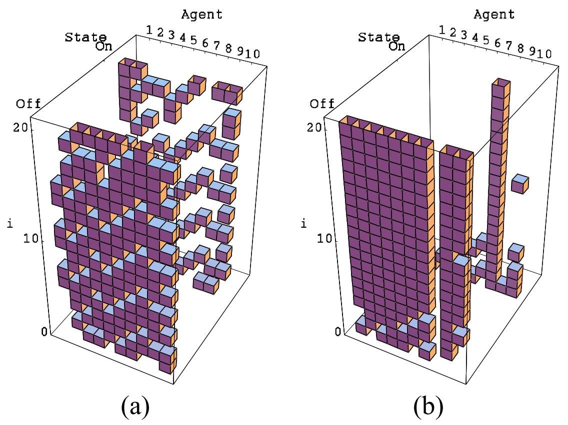


Fig. 15 . Evolution of the system in (a) instable conditions and (b) with locking.

As it can be seen from the MDM in Fig. 15 (a) the system is unstable, with some perturbations, which is consistent with Fig 13. In Fig 15 (b), the system is stable, with agent 8 being on most of the time; this can be represented as (0,0,0,0,0,0,0,1,0,0) –or 4 in decimal representation- which is consistent with Fig. 14.

In Fig. 16 an Interaction Network with 9 agents is shown. The topology is defined as {{1, 2}, {2, 3}, {2, 4}, {3, 1}, {3, 9}, {4, 5}, {5, 4}, {6, 7}, { 7, 8}, {8, 9}, {9, 6}}. There are 3 cycles {{8, 9, 6, 7, 8}, {5, 4, 5}, {3, 1, 2, 3}}, and the rules are {0, 0, 1, 1, 1, 0, 1, 1, 1}. For each cycle the fathers-descendant list is {{8, 0}, {9, 0}, {6, 0}, {7, 0}}, {{5, 0}, {4, 0}} and {{3, 4}, {1, 0}, {2, 2}} respectively, and therefore the locking vector is {0, 1, 1, 1, 0, 1, 1, 0, 1}, locking nodes 8, 5 and 1. The system shows cyclic behaviour without locking (see Fig. 17). In Fig. 18 the behaviour of the system, with the locking, is shown.

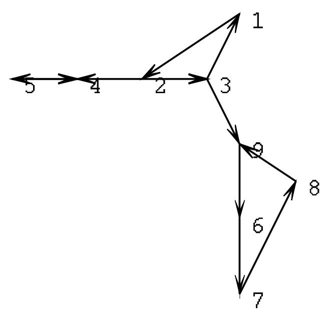


Fig. 16. Interaction Network with 9 nodes and three cycles.

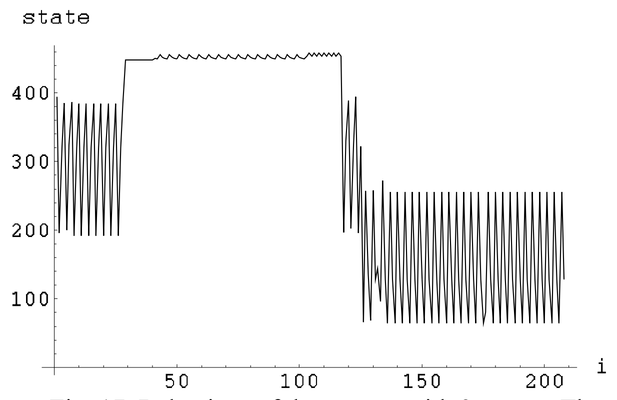


Fig. 17. Behaviour of the system with 9 agents. The system is oscillating.

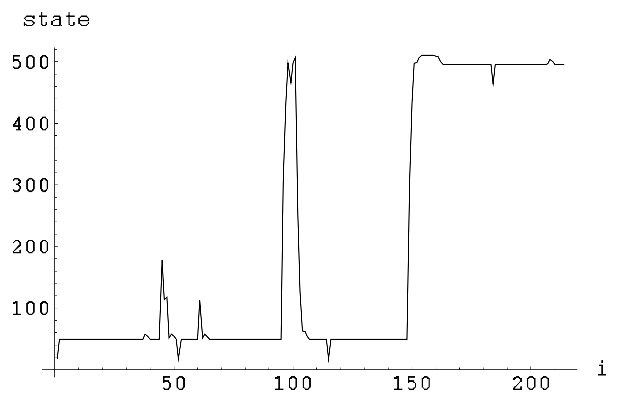


Fig. 18. Behaviour of the system with 9 agents. The instabilities have been removed by the locking mechanism.

The MDM representation can be seen in Fig. 19. (a) shows the instabilities, with 2 periodic patterns. (b) shows a static system.

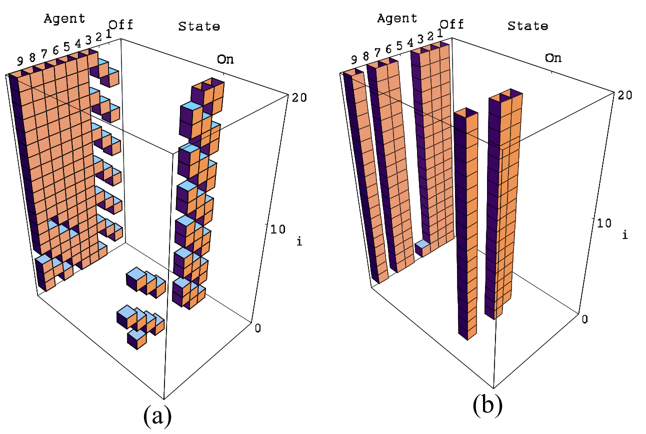


Fig. 19 . Evolution of the system showing (a) instable conditions and (b) with locking.

5.2 Testing with Emulated Devices

We implemented the locking method with a UPnP (Universal Plug and Play) system using the Siemens Java SDK for UPnP technologies [33]. An important difference to the simulated experiments is that this network included both delays (eg propagation, stack handling etc) and user interaction (eg turning lights on off) that provides a more accurate refection of a real environment. Thus, every device (lights in this case) has a user interface which allows the user to turn it on and off.

5.2. 1 UPnP Testbed

The iDorm is a multipurpose space, taking the form of a domestic apartment, with areas for varied activities such as sleeping, working and entertaining. It is populated with numerous networked embedded sensors (temperature, occupancy, humidity, light level) thereby making it possible to create and control systems such as those for entertainment, security, energy efficiency, care and work by orchestrating the coordination of the networked devices. It is based around three wired networks, Lonworks, 1-wire (TINI) and IP plus two wireless networks; WiFi & Bluetooth [23]. Universal Plug and Play (UPnP) is used as the common interface (middleware) to the iDorm, enabling automatic discovery and configuration. Our system was built on top of the low level UPnP control architecture, enabling it to communicate with the UPnP devices and orchestrate their action in the iDorm (in our tests we used a real network and emulated devices).

In order to test this approach, we used 3 scenarios:

Scenario I: The topology is coded as {{1,2},{2,3},{3,2},{4,3}} (see Fig. 20). This shows, there is one cycle involving agents 3 and 2. The rules of interaction are {0,1,1,0}, where 1 is an AND gate, and 0 is an OR gate. In Fig. 21 there is a screenshot of the system running with 4 devices.

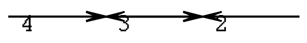
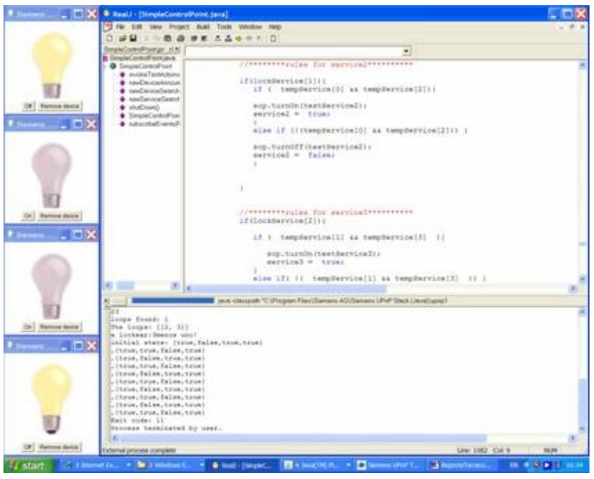


Fig. 20. Interaction Network (IN) for scenario I.



```
//*********rules for service2**********

if(lockService[1]){
   if (  tempService[0] && tempService[2]){

   scp.turnOn(testService2);
   service2 =  true;
   }
   else if (!(tempService[0] && tempService[2])) {

   scp.turnOff(testService2);
   service2 =  false;
   }


}


//*********rules for service3**********
if(lockService[2]){

   if (  tempService[1] && tempService[3]  ){

      scp.turnOn(testService3);
      service3 =  true;
   }
   else if( !(  tempService[1] && tempService[3]  )) {
```

```
java -classpath "C:\Program Files\Siemens AG\Siemens UPnP Stack (Java)\upnp\"
{}
Loops Found: 1
The Loops: {{2, 3}}
s locker: Remove uno!
initial state: {true,false,true,true}
,{true,true,false,true}
,{true,false,true,true}
,{true,true,false,true}
,{true,false,true,true}
,{true,true,false,true}
,{true,false,true,true}
,{true,true,false,true}
,{true,false,true,true}
,{true,true,false,true}
Exit code: 1
Process terminated by user.
```

Fig. 21. Screenshot of the system running 4 devices (lights in this case). The graphics mirror how the lights change according to the rules, and the states (ie on/off).

Fig. 22 shows the system behaviour in which the oscillations are dependent both on the rules and the initial state.

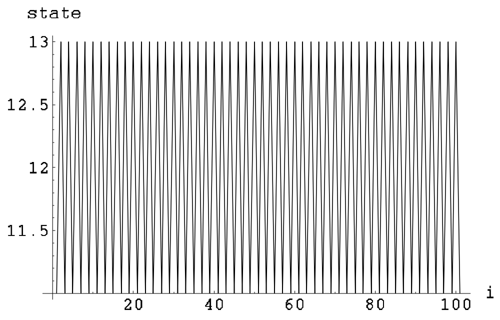


Fig. 22. Evolution of the system with 4 agents.

When locking is enabled, the oscillations are clearly prevented, leading the system to a stable state (unless the user alters the system), as is shown in Fig. 23.

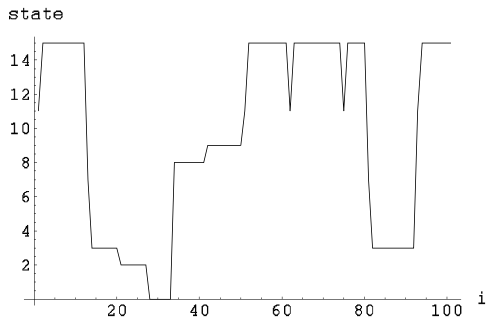


Fig. 23. The evolution of the system when subject to locking and perturbations from the user.

Fig. 24 The MDM showing the evolution of the system from iteration 0 to 20. When the locking is applied, agents 3 and 4 remain on.

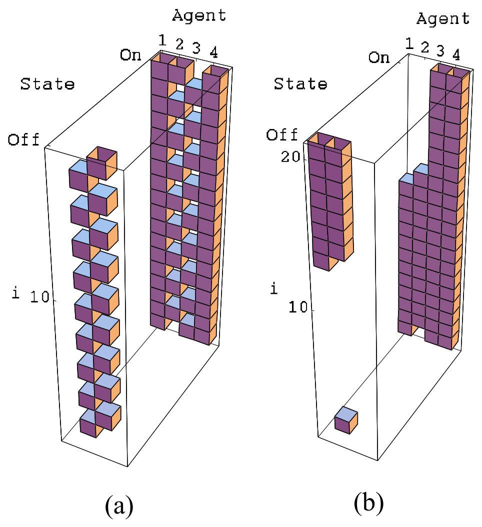


Fig. 24 . Evolution of the system in (a) instable conditions and (b) with locking.

Scenario II: Here there are 5 agents with an Interaction Network topology {{1,2},{2,3},{3,4},{4,2},{5,4}} (see Fig. 24). In this case there is a cycle involving 3 agents (2, 4 and 3). The rules of interaction are (1,0,1,0,1). These rules can lead the system to an instable state, as illustrated in Fig. 26. Instability was prevented successfully via locking (see Fig. 27).

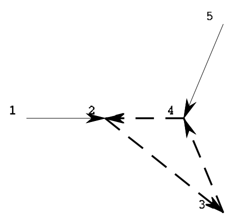


Fig. 25. Topology of the Interaction Network for 5 agents. In dashed lines a cycle involving agents 2, 3 and 4 can be seen.

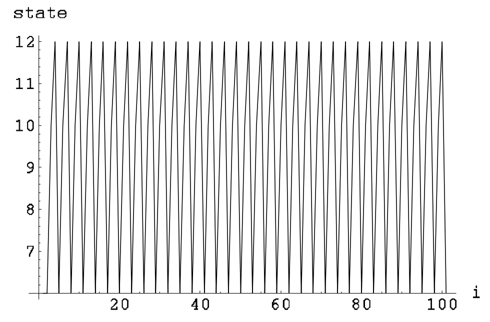


Fig. 26. Evolution of the system with 5 agents.

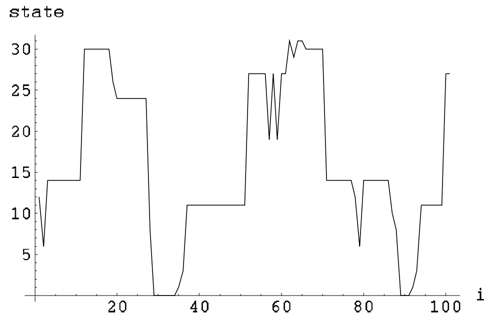


Fig. 27. Response of the system with 5 agents. The instabilities have been prevented.

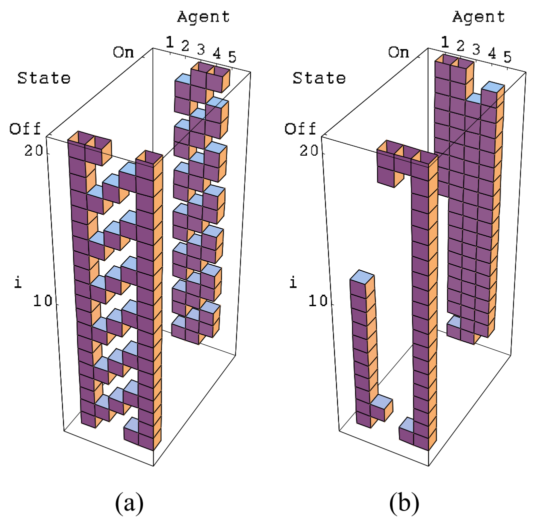


Fig. 28. MDM of the system with 5 agents showing (a) instable conditions and (b) with locking.

Scenario III: This tested the system for 7 agents. The topology of the Interaction Network is {{1,2}, {2,3}, {3,4}, {4,6}, {5,4}, {6,2}, {7,6}} (see Fig. 29). There is one loop involving agents 2, 3, 4 and 6. The rules of interaction are {1,0,0,1,1,1,1}. With these rules, instability can be present. However, when the locking mechanism is on, oscillations are prevented (see Fig. 30).

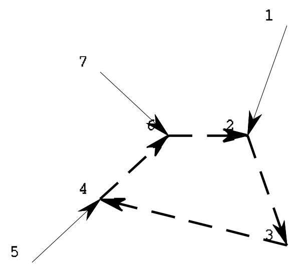


Fig. 29. The Interaction Network showing the dependencies of 7 agents. The dashed lines depict a cycle.

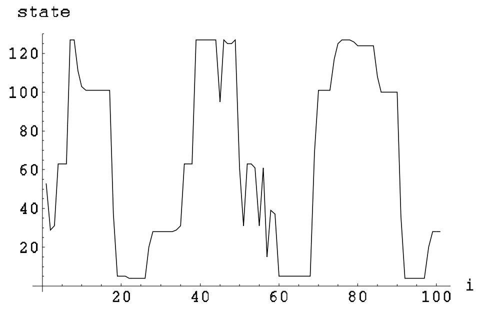
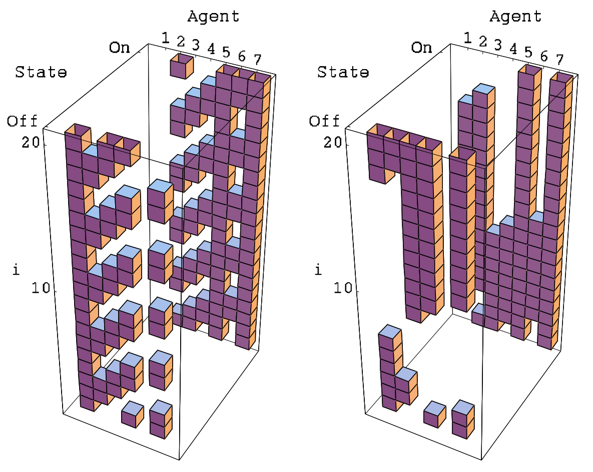


Fig. 30. Response of the system with 7 agents. When the locking mechanism is on, the oscillations are prevented.



(a) (b)

Fig. 31. MDM of the system with 7 agents. showing (a) the system in an instable condition and (b) in a stable condition (with locking).

5.3 Results Discussion

We have implemented and tested our strategy for finding and eliminating instability both using computer simulations and emulations based on a UPnP implementation that used a Siemens Java SDK. In the first case there were three interaction networks, with 5, 7 and 10 nodes respectively (see Figs. 5, 8 and Fig. 12) and randomly generated rules of interaction, together with random perturbations emulating the user interaction with the system. The IN with no loops didn't show any periodic behaviour. The systems with one and two loops in the IN associated showed instability (see Fig. 9 and Fig. 13); however, when the locking mechanism is implemented (in the first case locking node 4, and in the second case locking nodes 4 and 8 (all of them minimizing the impact on the network), the instability is removed (see Fig. 10 and Fig. 14).

In the UPnP implementation, we utilised 3 scenarios, with 4, 5 and 7 agents, using Boolean rules of interaction (AND and OR). In all the cases a closed loop was present. The user was able to interact with the system (turning on and off the lights). Without the node locking, cyclic instability can be observed (see Fig. 22 and Fig. 26). However, with locking the cyclic behaviour was prevented (see Figs 23, 27 and 30). These results are encouraging, even with the preliminary UPnP implementation, as the computer simulations have shown the locking to be effective on much larger and complex topologies, such as the one shown in Fig. 12, with two overlapping loops (sharing node 5).

The Multi-Dimensional Model MDM, representing the agent, its state and the evolution over time (in this case, iteration) was used in all the cases. The MDM has proven to provide a very useful representation, consistent with the decimal representation of the binary state of the system. The MDM can show, not only periodic and stable behaviour, but it is possible to visualize, at great detail, the behaviour or state of every single device present in the environment.

6. Interaction Networks and its Applications in Other Domains

Multiagent systems try to mimic capabilities such as reasoning, planning and learning [6] as seen in nature and society, being a general metaphor of the living world.

Distributed systems, with complex interrelationships such as companies, countries, economies, society, automatic trading systems and culture are susceptible to be modelled using multi-agents with complex and time dependant rules and dynamic interconnections [13]. Thus multiagent systems provide a useful tool to analyse and represent our world as a complex socio-technical system [10], [15], [16]. Work in this direction has been done, trying to analyze and destabilize terrorist networks removing leaders in the groups [10]. In this domain, the presence of loops in the network could suggest redundant leadership; our approach offers a way to analyze and reason about this problem, exposing redundant leaders in a given organization.

Economic behaviour of international companies in complex global markets, try to reason and learn not only from themselves but from their competitors. They receive feedback from their customers and suppliers, and coordinate their actions and strategies to achieve common goals, using very well defined rules and mimicking (or modifying) the behaviour of other participants [2], and under proper conditions could show instable behaviour. In share trading, business strategy and global enterprises (including governments) the behaviour of others is a key factor, and our work offers a tool to explain analyze and suppress –when appropriate –cyclic behaviour.

In this paper we have argued that, capturing relationships of dependency is very important for pervasive computing, in order to prevent unwanted outputs or even unstable cycles.

Knowledge networks (who-knows-what), information networks (what ideas are related to what), assignment networks (who is doing what) and social networks will gain more presence and importance in our complex world [10], [15],[16]. Interaction Networks are useful tool to represent complex interrelations between rule-based autonomous and coordinating agents (or equivalent entities), exposing loops in the interrelationships between the participants that could lead the system to cyclic instabilities.

In this section we have make an analogy between synthetic agents and social and organizational systems. Our next step will be to apply our Interaction Network Theory in order to reproduce some of the well-known cyclic behaviour in social and organization based systems

7. Conclusions and Future Work

In this paper we have described a challenge to achieving the vision for ambient intelligence; how to overcome cyclic instability in coordinating multi agent systems. As pervasive computing paradigms, such as ambient intelligence, utilise systems of interdependent agents, we contend that such behaviour represents a significant obstacle to the commercial exploitation of this technology.

Fig. 32. iDorm of the University of Essex.

In a bid to address this challenge we have devised a method of defeating instability in networks of coordinating pervasive computing devices that we call INPRES (Instability Prevention System). At the heart of this are three algorithms, one to define the process of eliminating instability, a second to identify closed loops and a third to select the nodes to be locked (both based on a measure of value and the users preferences). As part of this work we have created a formal framework for describing and reasoning about the problem (Interaction Networks), a visualisation model (Multi-Dimensional Model - MDM) and have proposed a methodology for overcoming the problem (Intelligent Locking). We have used both simulation and real devices to show the effectiveness of our methods. Simulation provide great flexibility, allowing experiments with arbitrary structures and sizes of networks (eg showing that the approach is scalable) whilst the experiments with actual devices has allowed us to see the effects of network and processing delays, together with user interaction. User interaction plays a fundamental role, which was not easy to see in computer simulations. For example, when the system has reached a stable state and the user interacts with one agent, it is possible to see the updates of the states in the neighbourhood, and when the locking mechanism is activated, how these changes are stopped, as the device that is locked prevents the propagation of the changes. On the other hand, the inclusion of sensors (light, movement, pressure, temperature, etc) will increase the complexity of the topology of the Interaction Network (but not the dynamic properties of the system!), as they cannot be part of a loop (they could only be fathers in the digraph), because its state depends on environmental conditions or user behaviour. Using these approaches we have shown that the locking mechanism is effective in the elimination of the unwanted cyclic behaviour, although the cost on the overall system is some temporary loss of functionality.

We have presented a theorem involving 2 agents, with arbitrary (but static) delays in the communication between them. We have shown that this system can have instabilities, with a period that depends on the delays. This delays are equivalent to multiple perturbations, preventing the system from evolving according to the rules; in our example, the system oscillated between (1,0) and (0,1) with a number of states (0,0) in the middle, depending on the delays suffered by each agent. As we have shown, our strategy prevented oscillations even in the presence of noise.

The Multidimensional Model represents graphically the set of agents, their states and time (or iteration), and has been shown to be a very valuable tool. It illustrates the devices involved in cyclic behaviour and the periods concerned. This model permits analysis and classification of different kind of perturbations. Some perturbations are weak and do not significantly destabilise the system whereas other perturbations are strong, changing dramatically the behaviour of the system. With this, we have developed a set of complimentary tools to analyse rule-based multi-agent systems: the MDM showing the evolution of the device-state plane in a very detailed way, the IN showing the functional dependencies of the agents, where a closed loop detection warns of possible instabilities, and the mapping from boolean to decimal state, showing the behaviour of the system and its evolution over time in a macroscopical way. Two representations: micro and macro evolution (MDM), plus a formal functional representation (IN) form the core of our contribution to the understanding and preventing instabilities in this complex pervasive computing environment.

As a future work we are planning to test our strategy with larger more complex topologies (in particular with multiple coupled loops) and with more complex rules. Also, as locking a node will impair, temporarily, some functionality of the system, the choice of what to lock and how long to lock (where there are options) is of some significance to the user. Thus a next step in our work is to experiment with a user based “locking preference” system. We will investigate both user specified preferences together with autonomous agent learning of preferences. For this we will run experiments in our test bed (iDorm[2] -a full size apartment that is fitted with pervasive computing technology and agents; see Fig. 16) in order to provide additional evidence of the strategy, and to refine the locking mechanism with information of the user’s preferences.

Finally, we are planning to extend our interaction network model to include other domains where instability can be

[2] http://iieg.essex.ac.uk/idorm

present, as it could provide useful tools and mechanisms to understand and remove cyclic behaviour in these areas. Although this is a challenging direction to our research, our results are encouraging and we look forward to reporting on our progress in future papers.

Acknowledgements: Victor Zamudio would like to acknowledge the support of the Mexican National Council for Science and Technology (CONACyT).

References

[1] H Abelson, D Allen, D Coore, C Hanson, E Rauch, G J Sussman and R Weiss. Amorphous Computing, Communications of the ACM, May 2000. Vol 43 No. 5.

[2] R L Axtell, Effects of Interaction Topology and Activation Regime in Several Multi-Agent Systems. Brooking Working paper, http://brook.edu/ES/dynamics/papers/interaction.

[3] S Baik, H Lee, S Lim and J Huh. Managing mechanism for service compatibility and interaction issues in context-aware ubiquitous home. International Conference on Consumer Electronics, ICCE. 8-12 Jan. 2005 page(s): 369 - 370.

[4] L R Brothers, E J Cameron Y J Lin and M E Nilson Elenita Silverstein. Feature Interaction Detection. IEEE International Conference on Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, Volume 3, Issue , 23-26 May 1993 Page(s):1553 – 1557.

[5] V Callaghan, M Colley, G Clarke and H Hagras, The Cognitive Disappearance of the Computer: Intelligent Artifacts and Embedded Agents, Proceedings of the i3 2001, workshop WS4 on Cognitive Versus Physical Disappearance, Porto, Portugal, April 2001.

[6] V Callaghan, M Colley, G Clarke and H Hagras, A Soft-Computing based Distributed Artificial Intelligence Architecture for Intelligent Buildings, In the book entitled “Soft Computing agents: New Trends for Designing Autonomous Systems”, in the International Series "Studies in Fuzziness and Soft Computing“, (Eds: V. Loia, S.Sessa), Springer Verlag, Volume 75, pp. 117-145, 2002

[7] V Callaghan, M Colley, H Hagras, J Chin, F Doctor and G Clarke, Programming iSpaces: A Tale of Two Paradigms, in iSpaces. Springer Verlag, 2005, Chapter 24.

[8] V Callaghan V, J Woods, S Fitz, T Dennis, H Hagras, M Colley and I Henning I, The Essex iDorm: A Testbed for Exploring Intelligent Energy Usage Technologies in the Home, Proceeding of the 3rd International Conference on Intelligent Green and Energy Efficient Building & New Technologies, International Convention Centre, Beijing China, 26th-28th March 2007.

[9] E J Cameron, N Griffeth, Y J Lin, M E Nilson, W K Schnure and H Velthuijsen. A Feature-Interaction Benchmark for IN and Beyond. Communications Magazine IEEE Vol 31, Issue 3, March 1993 Page(s) 64-69.

[10] K M Carley, J S Lee and D Krackhardt, Destabilizing Networks. Connections 24(3);79-92. 2002.

[11] K Cheverst, N Davies, K Mitchell and C Efstratiou, Using Context as a Crystal Ball: Rewards and Pitfalls. Personal and Ubiquitous Computing, Vol 5, Issue 1. Feb. 2001, pp 8-11.

[12] J S Chin, V Callaghan and G Clarke, An End-User Programming Paradigm for Pervasive Computing Applications, The IEEE International Conference on Pervasive Services, Lyon, France, June 26-29, 2006 Page(s):325 – 328.

[13] G Clarke and V Callaghan, Ubiquitous Computing, Informatization, Urban Structures and Density, Built Environment Journal, Vol. 33, No 2 2007

[14] G Coulouris, J Dollimore and T Kindberg, Distributed Systems, Concepts and Design. 4th Ed. Addison Wesley. 2005.

[15] J E Doran, Simulating Societies using Distributed Artificial Intelligence. In Social Science Microsimulation (eds. Troitzsch K G, Mueller U, Gilbert G N and Doran J E). Springer: Berlin. October 1996. pp. 381-393.

[16] J E Doran, Iruba: An Agent-Based Model of the Guerrilla War Process. Presented at ESSA 2005 Conference, held in Koblenz, September 2005, and published in Pre-Proceedings.

[17] D Estrin, D Culler, K Pister and G Sukhatme, Connecting the physical world with pervasive networks. IEEE on Pervasive Computing, Jan-March 2002, Volume: 1, Issue: 1, pages: 59-69

[18] G Haggard, J Schlipf and S Whitesides, Discrete Mathematics for Computer Science. Thomson 2006

[19] H Hagras, V Callaghan, M Colley, G Clarke, A Pounds-Cornish and H Duman, Creating an ambient-intelligence environment using embedded agents. IEEE on Intelligent Systems, Volume 19, Issue 6, Nov-Dec 2004 Page(s):12 – 20

[20] K Z Haig, K L Kiff, J Myers, V Guralnik, C W Geib, J Phelps, and T Wagner, The Independent LifeStyle Assistant (ILSA): AI Lessons Learned. In the Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04), July 25-29, 2004. San Jose CA. pp 852, 857.

[21] J Y Halpern and R Fagin, A formal model of knowledge, action, and communication in distributed systems: preliminary report. In Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing (Minaki, Ontario, Canada). PODC '85. ACM Press, New York, NY, 224-236.

[22] K Henricksen, J Indulska and A Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. Pervasive Computing: Proceedings of the First International Conference Pervasive 2002, Zurich, Switzerland, pp. 167-180. August 26-28 2002.

[23] A Holmes, H Duman and A Pounds-Cornish. The iDorm: Gateway to Heterogeneous Networking Environments.

Proc. Int'l Test and Evaluation Association (ITEA) Workshop Virtual Home Environments, ITEA Press, 2002, pp. 30-37.

[24] M Kolberg, E Magill, D Marples and S Tsang, Feature interactions in services for Internet personal appliances. IEEE International Conference on Communications, 2002. ICC 2002. Volume 4, 28 April-2 May 2002 Page(s):2613 – 2618

[25] D Lewin, Logical Design of Switching Circuits. 2nd Ed. Nelson. 1980.

[26] J M Martins Ferreira, T Amaral, D Santos, A Agiannidis and M Edge, The Custodian Tool: Simple Design of Home Automation Systems for People with Special Needs. Presented at the EIB Scientific Conference. Munich, October 2000.

[27] M Mowbray and M Williamson, Resilience for Autonomous Agents. Technical Report HPL-2003-210. Internet Systems and Storage Laboratory, HP Laboratories Bristol. October 17th 2003.

[28] S Pemmaraju and S Skiena, Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica™. Cambridge University Press 2003.

[29] S Qutub, R Alami and F Ingrand, How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'97., Volume 3, Grenoble, France 7-11 Sep 1997 Page(s):1610 – 1615

[30] P Ramagnino and G L Foresti, Ambient Intelligence: A New Multidisciplinary Paradigm. Systems, Man and Cybernetics, Part A, IEEE Transaction on. Jan. 2005. Volume 35, Issue 1.

[31] M A Razzaque, S Dobson and P Nixon, Categorisation and modelling of quality in context information. In Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications. Roy Sterrit, Simon Dobson and Mikhail Smirnov (ed). 2005.

[32] M Satyanarayanan, Pervasive Computing: Vision and Challenges. IEEE Personal Communications, Vol: 8 Issue: 4, August 2001, Pages, 10-17.

[33] UPnP Forum: http://www.upnp.org/

[34] S H Unger, Hazards, critical races and metastability. IEEE Transaction on Computers, Vol. 44, Issue 6. June 1995, pp 754-768.

[35] D J Watts and S H Strogatz, Collective Dynamics of Small-World Networks. Nature, Vol 393. June 4 1998.

[36] G Weisbuch, Complex Systems. Lecture Notes Volume II. Santa Fe Institute Studies in the Sciences of Complexity. 1991.

[37] M Wilson and E Magill, A Model for Service Interaction Avoidance in Home Networks. In Proc. Of the 5th Annual Postgraduate Symposium on the Converge of Telecommunications, Networking and Broadcasting. Liverpool. June 2005.

[38] M Wilson, E Magill and M Colberg, An Online Approach for the Service Interaction Problem in Home Automation. Consumer Communications and Networking Conference 2005 CCNC Second IEEE, 3-6, Jan. 2005, pp. 251-256.

[39] S Wolfram, The Mathematica Book, 5th ed. Wolfram Media, 2003

[40] V Zamudio, V Callaghan and J Chin, A Multi-Dimensional Model for Task Representation and Allocation in Intelligent Environments Proceedings of The Second International Symposium on Ubiquitious Intelligence and Smart Worlds (UISW2005). Nagasaki, Japan. December 6-7, 2005.

[41] V Zamudio and V Callaghan. Unwanted Periodic Behaviour in Pervasive Computing Environments, The IEEE International Conference on Pervasive Services ICPS06, Lyon, France, 26-29 June 2006

Author Bios

Victor Zamudio holds a MSc. in Computer Science from Monterrey Tech (Mexico) and a B.Sc. in Physics. He has a strong interest in pervasive computing, multi-agent systems and ambient intelligence.

Vic Callaghan holds a Ph.D in Computing and B.Eng in Electronic Engineering from the University of Sheffield. He is Professor of Computer Science at Essex University. He has contributed to over 100 papers journals, conferences and books and currently, he leads the Inhabited Intelligent Environments Group (IIEG) and is director of the Digital Lifestyles Centre.