Building Ambient Intelligence into a Ubiquitous Computing Management System

Julie A. McCann, Peter Kristofferson, Eduardo Alonso

***Abstract*—The ANS is a ubiquitous computing management tool, designed to mimic the Autonomic Nervous System of living creatures. Its job is to manage an intelligent distributed environment for e-medicine applications. The homes' computing power consists of many computing units of varying capacities (e.g. sensors). To seamlessly plug/unplug units into the environment requires that the system have a high degree of adaptability and reconfigurability since users cannot perform systems management. This paper describes the ANS and how the idea of Intelligent Modelling can aid self-optimisation of the system structure required to drive such an ambient system.**

***Index Terms*—Self-regulating, autonomic, ubiquitous computing, and agent-based remodelling.**

I. Introduction

Recently we have seen a paradigm shift in computing towards pervasive and ubiquitous computing. An example application of this technology is the *intelligent home* where '*the system*' is the homes' complete computing power and which consists of many computing units of varying capacities and basic functionality. Together these sensors collect data, filter this data, and collaborate with each other to provide information on the state of the environment. Where the Intelligent home is sustaining the life of a heart patient this aspect of the ubiquitous system becomes much more important. Here the system's primary goal is to keep the patient alive and 'happy', while doing so in an efficient and robust manner. Therefore this environment requires that the system have a high degree of automatic adaptability and reconfigurability since it is inconceivable that users would perform explicit systems management and maintenance. That is, we must be able to seamlessly plug and unplug units into '*the system'*. The functionality of these "ubiunits" would be determined by the context into which they are plugged or located. For example a *webpad* unit can be the remote control for your television when in the sitting room, the interface to your fridge in the kitchen or just idly hung on a the wall when not needed; possibly doing backups of system data to the network or ordering groceries to stock a depleted refrigerator. When a unit breaks or upgrades are required, new units can also be added seamlessly or when a doctor enters the house, it recognises his/her PDA as a 'friendly alien device'.

Essentially the ANS (Autonomic Networked System) is a ubiquitous computing management tool, which is designed to mimic the ANS (Autonomic Nervous System) of living creatures. The organic ANS is the part of the nervous system controlling many organs and muscles within the body. However, we are unaware of the workings of the organic ANS because it functions in an involuntary, reflexive manner. What is particular about the organic ANS is that it is flexible, constantly in operation and that it happens in the background without our interference or knowledge of its mechanism. This is exactly what is needed to support ubiquitous computing environments, especially in the application of the 'intelligent home' and medical applications where constant technical support is impossible. Further, such a system should provide the 'intelligence' to optimise its operation through constant monitoring and tuning to achieve its goal.

Currently, we do not visualise our computing technology to be disposable, such that we can throw away a faulty unit and replace it as easily as we would a light bulb. However this is exactly the paradigm required to achieve Weiser's vision of calm in pervasive and ubiquitous computing [2].

II. Scenario

An example environment we would foresee the ANS supporting is the home, which consists of a set of sensors, and these sensors collaborate to various degrees to monitor and support a patient who has already had a heart attack but is on medication awaiting heart by-pass surgery. Here we have a combination of programmable logic controllers that look after the opening and closing of windows, lighting, and temperature of the room etc. Furthermore we have generic sensor devices like that of Motes [6]. A mote is a tiny wireless board with an onboard processor that runs operating system (TinyOS) code, a two-way ISM band radio transceiver, and a logger memory capable of storing up to 100,000 measurements. Measurements can come from sensors, which deliver vision, acoustic, pressure etc data. They are positioned around the home and together monitor the context of the patient. What is

Manuscript received November 19, 2003. (Write the date on which you submitted your paper for review.) This work was supported in part by the Department of Trade and Industry, United Kingdom as well as the Dean's award, City University, London.

Julie A. McCann is with the Department of Computing, Imperial College, London, UK (phone: +44 2072530845, email jamm@doc.ic.ac.uk)

Peter Kristofferson is with the Department of Computing, City University, London, UK (email dn184@soi.city.ac.uk)

Eduardo Alonso is with the Department of Computing, City University, London, UK (email eduardo@soi.city.ac.uk)

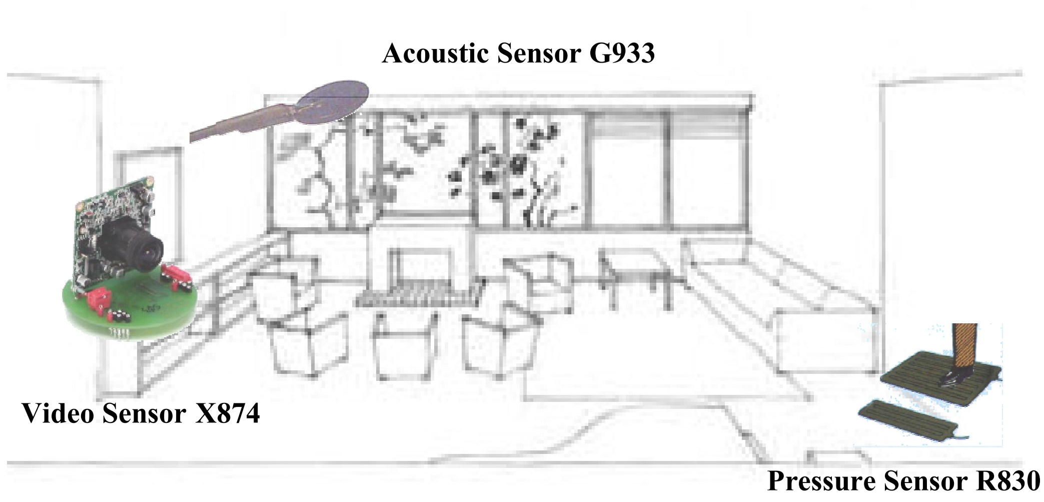

Figure 1 Video, Floor and Acoustic Sensor in the Intelligent Home

interesting about these modern sensor systems is their capacity to process or filter data or work on the behalf of another sensor.

A patient has a heart sensor embedded within them and this is communicating via a wireless link to a PDA that is carried with them. The PDA relays filtered heart data to the ANS. A vision detector is trained on the patient and is capturing vision data and processing it to give the ANS an idea of the position of the patient and whether or not he/she is standing, sitting or lying down. Other sensors capture information that helps with deciding the physical context of the patient. For example there is a pressure sensor on the floor that shows where the person is in a given room. There could be a situation where the heart monitor has detected that the patient's heart is beating dangerously fast. This can be due to many causes, one of which is that the patient is exercising or is walking upstairs (i.e. his/her context). However if the patient's context is that they are sitting in a chair this may indicate there is a problem. In this instance the ANS contacts both the doctor and a predefined relative or neighbour via the Internet. If the medical logic within the ANS leads it to believe that the patient is in grave danger it may then use the implantable defibrillator to send a shock to the patient to restart the heart. Likewise, when the patient leaves the house, the PDA uses mobile phone technology to keep in constant communication with the home sending patient data back for analysis.

Therefore essentially the ANS can be viewed at two levels, medical (application) logic to monitor and understand the status of the patient as a given time and self (architectural) logic, that allows the ANS to reflect on how well it does its job. It is the self-knowledge that we are focusing on in this project. This is what we mean by ambient intelligence or autonomicity.

For illustrative purposes, the scenario we present in our paper is thus. There could be a situation where a sensor fails and the system has to gracefully reconfigure to cope. For example the floor sensor has been accidentally kicked and is no longer sending information to the ANS, however context data is still required. Therefore the ANS may require that the data from the vision sensors are obtained and corroborated with perhaps another sensor to take the pressure sensor's place (e.g. acoustic sensors). Alternatively the vision sensor must change its processing to cope (i.e. stream more data of better quality to indicate the position of the person).

III. Intelligent Modeling

Effectively the ANS system will have to rearrange its structure to cope with the changes in the environment. The higher goal of the system is to keep the patient alive and it will strive to achieve this by reconfiguring, either through reactive or proactive actions. This is achieved by using an intelligent model. A model is a simplified representation of a problem or scenario or reality. We define modeling as the process of changing that model with a specific goal in mind. An intelligent model has the capacity to change its own structure, to remodel itself, with a specific goal in mind. For example this could be structural change through the combination of different parts of the model to form a new whole.

Each node (e.g. processing element with a sensor) has a set of abilities that pertain to that node and its role within the system. The nodes are primarily autonomous and do not have global knowledge. By that we mean that the nodes do not know the overall structure of the model but have limited knowledge about a set of neighbouring nodes. However they do have an understanding of the rules of the model and its ultimate goal. What makes an intelligent model different from any other similar system is the fact that in certain cases operations, which are not normally allowed to be performed by the node according to the rules of engagement inside the model, can be performed as long as the result is coherent according to the rules of the model. The reason behind this ability is the same as in human based modeling; the result should be beneficial to the model as a whole. The power to achieve this can be seen as a higher-level management of the node and is held in the logic of both the application and architecture.

The abilities that enable the node to reason about its own position in the environment and how to improve on that situation are:

A. Reasoning

Each node has the ability to carry out simple reasoning about its current situation, this may take the form of overcoming a lack in information because, for example a information source provider dies. This may entail helping another node to find a suitable information substitute, or reasoning about new ways of acquiring information, or other routing or rerouting capabilities, etc. This can also be Pro-active, where the system is able to predict a problem and can planning for the future or preparing for possible changes.

B. Communication and Finding out information

It needs the ability to communicate with other nodes, to formulate questions to find out information about specific problems and to be able to ascertain the state of a neighbouring node by probing their knowledge.

C. Changing information suppliers / consumers

It needs the ability to change information suppliers connected and neighbouring suppliers of knowledge to non-connected neighbouring nodes. This holds for a node, which provides knowledge also routing

It needs the ability to route information on behalf of other nodes. This can be used to perform fault tolerance etc.

D. Knowledge

Each node declares its service and functionality in a standard way that all nodes in the system understand. Furthermore it describes the data that it provides, its format and alternatives that it knows the can do a similar job, transformations describing how the data can be transferred to other data or how to derive info from the existing data. This can be implemented through the use of ontology.

In other words the nodes display the following features:

- Are autonomous
- Posses social skills
- Are reactive
- Are proactive
- Have ability to affect their environment

These are also the features of agents as described by Wooldridge et al [4]. What enables an intelligent model to work is the behaviour of agent like entities that represent the most basic parts of the system – the model. An intelligent model should be considered as a whole even though it is distributed. It can be seen as a representation of a scenario with the ability to change itself according to a specific goal and thus create a new representation without human intervention. When applied, this idea will enable a system to reconfigure itself. However rather than using economy, or negotiation, as a reconfiguration tool this system will be using automated modeling. The idea of intelligent model seems to be similar to the ideas of social rules in the agent community but only superficially [3] [5]. In social rules based systems the agents can break the rules to achieve their personal goals but will have to pay the penalty for doing so. In an intelligent model an activity that does not conform to the rules of engagement can be performed only if the result is correct according to the global rules of the model. However a penalty is not paid for doing so. In a way this can be seen as defining the emergent property of the system in advance.

IV. How Intelligent Modelling can be used in the ANS system

Each node (sensor) in the home network has two abstract levels of system logic; Application Logic and Architecture Logic. However these both interact to allow the system to adapt at runtime to cope with the changing environment. Each node in the system contains an operating system (e.g. TinyOS) and its own element of the ANS. It is assumed that each node is autonomous and that there is no central control element. This allows the system to be scalable, flexible and robust in that there is no central area of failure.

The software on each node is component based and can self-(re)configure with the help from monitors, which provide environmental data (e.g. current performance ratings etc). The whole system is viewed as a component-based system that further can reconfigure using the Intelligent Model principle. The initial architecture consists of a closed adaptive architecture that implements closed adaptivity that is reactive. However the model is structured in such a way that proactive activity can be sustained.

The active configuration of a component is monitored, i.e. is fed information from monitors or *gauges* (which aggregate raw monitor data for more lightweight processing). The monitors can be monitoring the effectiveness of the individual components, their interaction or the interaction outside the system. Should the system detect a problem, or some form of deteriorated quality of service (regarding the overriding goal

Node ID: G933
Name: Acoustic Sensor
Functionality: Primary< Supply Sound data, Supply Voice Data>
Secondary <Supply Location Data>
Communication: Supply Request, Sound, Voice, Location
Neighbour: R830, Idr3956

Sound raw type::= bitstream (lowrate)
Sound output_type::= sound < sound (bitstream) >

Voice raw type::= bitstream (midrate)
Voice output_type::= voice <voice (bitstream, natural language)>

Location raw type::= bitstream (highrate)
Location output_type::= location <location (bitstream, xy_coord)>

<Supply Request (supply, source)>

Figure 2. Acoustic Sensor Logic

of the system) then the system decides to reconfigure.

For example, using the scenario described above. If the system should detect that the floor sensor has gone out of operation then it needs to cope with this change. The notification of the sensor dying is effected through the other components that use its information to gather context as to the position of the patient at a given time. Those sensors or components have a set of neighbours known to them. They contact these neighbours to see if they have knowledge of the missing component, its service, data and alternatives to supply that same data. In this example the floor component may be supplying the vision component information to target the position of the camera to determine the physical position of the patient better. Now that it has died, the vision component knows that there is an alternative sensor that might help; in this example it is the acoustic sensor component. During normal operation the acoustic component is monitoring the general sounds of a given space, presenting voice to the natural language processor for audio commands from the patient for example. However when the floor sensor dies the acoustic component can become more sound sensitive to monitor where the major sounds that are changing (indicating the person moving etc) allowing the vision sensors to track the user better. The three sensors are presented in Figure 1. The information in the vision sensor's knowledge allows it to know what the floor data was providing and knew that the acoustic sensor can provide similar data if provided at a higher quality. The sensor logic for the three sensors is presented in figures 2, 3 and 4 respectively.

The data in figures 2,3, and 4 convey the description of each of the nodes in our scenario. For example Acoustic Sensor (figure 2) is given a unique identifier and a description that has meaning in the ontology used by our medical ANS system. The functionality can either be primary functionality or secondary. This means that under normal operation the sensor will carryout the primary functionality. Secondary functionality indicates it can help if another node should fail. Note that it doesn't need to know of other nodes, just describe its functionality appropriately. That is, the Acoustic Sensor's primary functionality is to record background sounds and the patient's voice. The background sound is recorded and not acted upon; hence the *Sound* function takes a low-rate bit stream and does nothing other than record it to the ANS disks. Of more interest is the *Voice* functionality. This takes a mid-rate bit stream and using the *Voice* function translates this into natural language input from the patient to the ANS, e.g. a command to activate the heating would get ANS to turn the heating on.

Node ID: R830
Name: Pressure Sensor
Functionality: Primary <Supply Location Data>
Communication: Supply, Location
Neighbour: G933, jn4394

Location raw type::= bitstream (midrate)
Location output_type::= location <location (bitstream, xy_coord)>

<Supply Request (supply, source)>

Figure 3. Pressure Sensor Logic

Node ID: x874
Name: Video Sensor
Functionality: Primary< Supply Motion data, Supply Position Data>

Communication: Supply Request, Movement, Situation, Location
Neighbour: R380 (Location), G933

Motion raw type::= bitstream (lowrate)
Motion output_type::= movement < movement (bitstream, Location) >

Position raw type::= bitstream (lowrate)
Position output_type::= situation <situation (bitstream, Location)>

<Supply Request (supply, source)>

Figure 4. Video Sensor Logic

The Acoustic Sensor has a secondary function of being able to do some sort of location calculation. This takes the same bit stream but at a higher sampling rate for more accuracy[1]. This is filtered using the *location* function and x-y co-ordinates are obtained to relate the location of the patient in that room.

The supply request is the function that runs should the system detect a problem. For example, the vision sensor (figure 4) requires that the pressure (floor) sensor (figure 3) supply *Location* information. This is indicated by listing that the floor sensor is one of its neighbours and supplies location information. This is fed into both the position and motion functions to position the camera to focus on the patient, calculating the *position* (i.e. whether the patient is standing up or sitting) or *movement* (i.e. the degree of movement).

The vision sensor needs to know where to look for alternative *Location* information should its stream of Location data fail. It does this by executing the supply request function. Figures 5 and 6 illustrate the general form of the supply request function and use Table 1 to make the remodelling logic more succinct.

The Vision sensor now needs Location data to position its camera correctly to carry out context processing. Figure 5 shows the logic enabling the system to reconfigure its operation, i.e. rules 1-3 are let the system initiate a search for an alternative resource. Once the sensor detects that Location data is no longer being sent it issues a request to its list of neighbours to ask if they supply location data (LD). The Vision sensor is the ORIGINATOR in this example. Its neighbours were consulted and then their subsequent neighbours in turn until either a response is returned or a timeout is reached. If a 'LD SUPPLIER unknown' is returned then it indicates that no other reachable node is able to supply the Vision Sensor with Location data. Here an event is signalled and a message is sent to the USER who could, for example, be the hospital that supplied the ANS and sensor kit

[1] Note when the higher sampling rate is used more of the sensor processing and power resource is used and therefore this should be used sparingly.

ORIGINATOR =the original node requesting rmation
SENDER = node that sent the message
NEIGHBOURES = all connected nodes
SUPPLIER = a node that supplies information
USER = person responsible for the system
LD = location data
NODE_X = the original node sending a statement

Table 1: Variables for Modeling Reconfiguration

to the home. If a node has been identified as one that can supply the data then a request is sent to route the data to the ORIGINATOR i.e. the Vision Sensor.

Figure 6 lists the logic held on other devices to help the ORIGINATOR satisfy its request. Rule 4 is activated when a request is received on a node for data. Here if it can satisfy it then it indicates this by sending a 'LD supplied by NODE_X' and initiating the stream of data. Otherwise it consults its list of neighbours if they can supply the data. Rule 5 allows the system to stop traversing the neighbourhood lists and indicate that there is no other reachable node able to help. In a large system it is unrealistic that all reachable neighbours will be traversed therefore the ORIGINATOR may put a timeout on the response whereby if reached the ORIGINATOR will send an error to the USER. Rule 6 indicated that the node is willing to supply data, otherwise it will route data on behalf of the other supplier to the ORIGINATOR.

Rule 7 of figure 6 is interesting in that the node being asked if it can route data but it has only enough processor resource to for its own function. However if a request comes through from a node, which has a more important status requiring location data, then that node knows it is for the betterment of the whole system to relinquish its processor and start supplying the ORIGINATOR with Location data (in the format and level of quality that is required). This is where the intelligent model is re-modeling the system as it operates.

More than one neighbour can return a 'LD supplied by NODE_X' message. A protocol is required which allows the ANS to make a decision as to which route to take. Initially a first implementation will go for the first neighbour to return with a positive message. This is good in that it indicates that it has the fastest route (which is a combination of node hops and bandwidth between nodes). More sophistic designs will inform the originator of a cost function (based on hops and network bandwidth at that time) so it can make a more intelligent decision.

```
1 IF LD not available
    ASK NEIGHBOURES "supply LD to ORIGINATOR"

2 IF RECEIVED MESSAGE == "LD SUPPLIER unknown" AND self == ORIGINATOR
    IF all NEIGHBOURES have replied AND LD still not available
        SEND warning to USER
3 IF RECEIVED MESSAGE == "LD supplied by NODE_X" AND LD still not available
ASK "LD supplied by NODE_X"'s SENDER "ROUT LD to ORIGINATOR"
```

Figure 5. Modelling rules: Node requesting missing information

V. Opening up the ANS model using the Intelligent Model

The initial ANS will be a closed adaptive model as all the logic and knowledge is prescribed and static. However for the system is to be able to evolve and cope better with new situations that the systems architects did not foresee, the system needs open adaptivity. This means that when a decision to reconfigure happens the system gets feedback on that reconfiguration.

For the system to be able to learn from and adapt to situations for which it was not originally designed the system needs open adaptivity. The system is continuously interacting with its surrounding environment in an ever-changing dynamic way. Each time the system receives an input; it executes an action that in turn affects the environment bringing it to a new state, that is, a new input. The problem in such scenarios is how to select, in a given state, the best policy so as to maximise the utility of the system taking into account the inherent uncertainty of the system-environment interaction. Reinforcement learning algorithms (see, e.g., [7]) solve this problem. Systems are rewarded at each state according to the designer's preferences learning in the long run which sequences of actions do adapt better to the environment in which the system is situated.

VI. Summary

Simply put, the resulting software emerging from this project will be designed in a component-based way. That is, the software's functionality is split into its elementary component parts so that functionality can be swapped in and out to reconfigure the overall architecture. Each software component consists of the code to carry out its function and a means to communicate how it does this to the other components in the system (we call this the semantic interface). This information is used when the overall system requires a new component to join it (i.e. the system will reconfigure) to allow it to choose the best component to do the job. The decision to reconfigure is made as a result of some monitoring component detecting change. Change can be in terms of a failed hardware unit or performance degradation. The aim of the system is to meet some form of Quality of Service (QoS) guarantees. When those guarantees cannot be met at a given time the system decides to reconfigure to try and overcome this. Consequently, our research carried to build such a system involves component-based software engineering and architecture modelling, knowledge of modern execution environments and tools, semantic interfacing and QoS definition and modelling [4]. Using the intelligent modelling idea the architecture becomes more open and self-adaptive. The ANS will be designed to integrate with the co-proposed projects in the Imperial College UbiCare Centre. Essentially the ANS will be concatenating the streamed data coming from the sensor systems developed by the UbiMon project (though the ANS project is not completely tied to these technologies).

References

[1] McCann J.A., Howlett P., Crane J.S., 'Kendra: Adaptive Internet System', Journal of Systems and Software, Elsevier Science, Volume 55, Issue 1, 5 November 2000 .pp 3-17.

[2] Weiser M., 'Some computer science issues in ubiquitous computing', Communications of the ACM, vol. 36, no. 7, pp 75-84, July 1993

[3] Will Briggs and Diane Cook, Flexible Social Laws, In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[4] Wooldridge M. and Jennings N. R. (1995): Intelligent agents: Theory and practice, *The Knowledge Engineering Review*, 10(2), pp. 115-152

[5] Alonso E. (2002), Rights for Multi-Agent Systems, d'Inverno M., Luck m., Fisher M., Preist C., *Foundations and applications of Multi-Agent Systems, UKMAS Workshops 1996-2000 Selected Papers,* LNAI 2403, Springer Verlag; p. 59-72.

[6] http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0048-03_A_MOTE-KIT.pdf

[7] Sutton R.S., and Barto A.G., Reinforcement Learning: An Introduction, The MIT Press, Cambridge: MA.

```
4 IF RECEIVED MESSAGE == "supply LD to ORIGINATOR"
      IF LD supplied by == SELF
      START supplying LD to ORIGINATOR
         ANSWER to SENDER "LD supplied by NODE_X"
      ELSE
      IF NEIGHBOURES other than the SENDER exist
         ASK NEIGHBOURES "supply LD to ORIGINATOR"
      ELSE
         REPLY "LD SUPPLIER unknown"

5 IF RECEIVED MESSAGE == "LD SUPPLIER unknown"
       IF all NEIGHBOURES other than the SENDER have replied AND all
                                replies are "LD SUPPLIER unknown"
            REPLY "LD SUPPLIER unknown"


6 IF RECEIVED MESSAGE == "ROUT LD to ORIGINATOR"
      IF LD used by == SELF
         START supplying LD to ORIGINATOR
      ELSE
         ASK NODE_X "SUPPLY LD via ROUTING"

7 IF RECEIVED MESSAGE == "SUPPLY LD via ROUTING"
      IF processing_resource is < MAX-threshold
         START supplying LD to SENDER
      ELSE if non-SUPPLY LD to ORIGINATOR == SYSTEM-GOAL
    BROKEN
           HALT current processing
           START supplying LD to ORIGINATOR
```

Figure 6. Modelling Rules: Other Nodes are consulted regarding a request for data supply