AMUSE: Autonomic Management of Ubiquitous e-Health Systems

N. Dulay[1], S. Heeps[2], E. Lupu[1], R. Mathur[1], O. Sharma[2], M. Sloman[1], J. Sventek[2]

[1]Department of Computing, Imperial College London
[2]Department of Computing Science, University of Glasgow
http://www.dcs.gla.ac.uk/amuse/

Abstract

Future e-Health systems will consist of low-power on-body wireless sensors attached to mobile users that interact with a ubiquitous computing environment. This kind of system needs to be able to configure itself with little or no user input, but more importantly, it is required to adapt autonomously to changes such as user movement, device failure, and the addition or loss of services. We propose the Self-Managed Cell architecture for such systems, and outline how the architecture supports an e-Health application in which on-body sensors are used to monitor a patient living in their home.

1 Introduction

Monitoring chronically ill patients as they go about their normal activity enables early release from hospitals and improves the patients' quality of life. Analysis and data mining of the monitored information can be used to predict potential problems such as a possible heart attack for a specific patient being monitored and to generate a warning to the patient or medical staff, but can also be used by medical researchers to understand the body changes that take place prior to a specific problem. The technology to enable 'healthcare everywhere', such as programmable body sensor nodes which use wireless communication to PDA/phones that interact with remote medical centres for logging and patient feedback, is now available. The UbiMon project is developing techniques for determining the conditions that indicate a medical problem for a patient and taking appropriate action to warn a patient or, if necessary, involve a medic in the monitoring loop. In addition, logging the data to servers allows data mining for medical research to determine typical patterns indicating medical problems – see [1] for further information.

On-body and environmental sensors may also be used in the home for monitoring elderly people to determine problem situations or deterioration of well-being over time [2]. However, configuration of the multiple sensors and software components that will form an adaptive body-area network or a home monitoring network is not currently feasible for non-technical patients or medics.

The Amuse project is developing autonomic management techniques for self-configuring and self-managing such systems. The systems must be able to add or remove components, cater for failed components and error prone sensors, automatically detect and adapt to a user's current activity and communication capability, as well as catering for interaction with health visitors or medics who attend patients or visit elderly people. User activity determination is critical, for example to distinguish between vigorous brushing of teeth and possibly a heart attack. Although we are using e-Health as a scenario, our aim is to develop generic techniques applicable to e-Science and other applications.

2 Self-Managed Cell

We advocate the concept of the Self-Managed Cell (SMC) as an approach for implementing autonomic ubiquitous systems. An SMC manages a set of components such as those in a body-area network, a room or even a large-scale distributed application. The components could be on-body sensor nodes, or in the future, "intelligent" implantable sensors and actuators (c.f. pacemakers and defibrillators) as well as smart phones and PDAs.

The main objective of an SMC is to support autonomic functions such as self-configuration, self-healing, self-optimisation. self-protection and context aware adaptation as appropriate. In the e-Health scenario described in section 3, patients or medics would not have the technical knowledge to configure the sensors for a patient. Instead the SMC should discover appropriate sensors and configure them to perform the appropriate monitoring.

There are several core services required for an SMC, which we currently assume are implemented on a PDA for a body-area network:

Event Bus: adaptive ubiquitous systems are essentially event driven as changes of state in resources need to be notified asynchronously to several, potentially unknown, recipients. An event may indicate discovery of a new component, component failure, change in context or medical condition e.g. ECG anomaly detected. We are implementing a simple publish-subscribe event system supporting at-most-once persistent event delivery in which the service attempts to deliver the event until it knows that the subscriber is no longer a member of the SMC.

Discovery service: it is essential to discover nearby components which are capable of being members of the SMC e.g. intelligent sensors, and other cells when they come into communication range. The discovery service interrogates the new devices to establish a profile describing the services they offer and then generates an event for other SMC components to use as appropriate. We have to cater for mobile wireless components which may drift in and out of communications range and distinguish this from permanent departure from the cell.

Role Service: roles provide a means of grouping components, and generally correspond to the *role* they play within the cell e.g. temperature sensors, heart sensors, context sensors, the patient, or a healthcare worker. Roles can be considered active components which receive discovery events and decide whether to assign components to a role which will govern what cell resources can be accessed and how the component interacts with other cell components. Roles are organised hierarchically and are written using pathnames (e.g. like file pathnames).

Policy service: policies provide the means of specifying the adaptation strategy for autonomic management. Based on the lessons learnt from our previous policy specification work [3] [4] [5] we are developing a new light-weight policy service appropriate for limited-resource devices. Authorisation policies specify what resources the components assigned to a role can access and obligation policies (event-condition-action rules) specify how roles react to events and interact with other roles. When a device is assigned to a role, the appropriate policies will be deployed to it. Policies can be added, removed, enabled and disabled to change the behaviour of cell components without reprogramming them. Policies also govern the behaviour of the discovery service, the role service and the policy service itself, allowing these to be tailored to specific applications.

More complex SMCs may support other services such as context services that provide information from the surrounding environment such as location, or from fixed infrastructure such as weather, pollen counts etc; security services that perform violation or attack detection, and support authentication and confidentiality; optimisation services which try to optimise performance according to a utility function etc.

3 e-Health Body Sensor Network Scenario

In this scenario, we are focusing on a body sensor network SMC monitoring a cardiac patient and how it interacts with a nurse who comes to visit the patient. The self-configuration function must first discover appropriate components and then configure them according to the role they perform in the cell. This would be triggered by assigning the discovered component to one or more roles, and could entail loading policies into an ‘intelligent’

```
1.  verify medic (c) → c.role="Nurse" and c.issuer="NursingCouncil"
    oblig  SMCdetect (n) → n.type="nurseSMC" and medic (n.credentials) ? /medics/nurses.assign (n.name, n)
2.  oblig  DeviceDetect (t) → t.type="tempsensor" ? /sensors/temperature.assign (t.name, t)
      ……….

3.  oblig every (mins(1))  → /services/discovery.findNewMembers()
4.  oblig every (mins(2))  → /services/discovery.pollMembers()
```

Figure 1 Example Policies

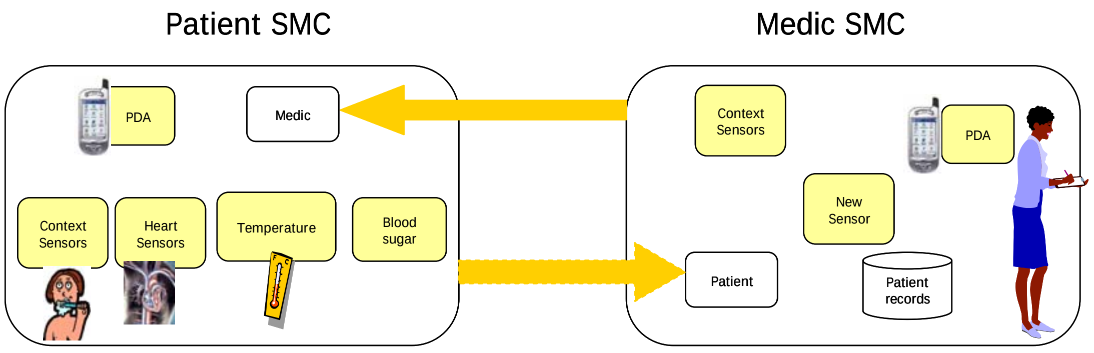

Figure 2 Peer to Peer Interactions Between SMCs

device or setting parameters on a simpler device. Each SMC's discovery service will periodically broadcast a request to discover new components. Any 'discoverable' component which is not a member of the cell responds with a message giving its identifier, address and device type. The device can then be queried to obtain its profile, which describes the services it implements, any credentials it offers etc. Components will have policies specifying how to respond to discovery requests. For example, a sensor component will only respond if it is not already a member of a cell, whereas a nurse might only respond to a patient whose identifier is in a preloaded patient list. Some example policies are shown in Figure 1.

This set of policies relate to device discovery, and specifies that a nurse (SMC) certified by the nursing council should be assigned to the medic role while temperature sensor devices should be assigned to the temperature role. Policy 3 indicates the frequency of polling for new devices and policy 4 specifies that devices in both roles should be polled to confirm that they are still reachable. Pollmembers is a role-specific action which maintains records of device presence and removes those devices that have been unreachable for some period of time, e.g.10 minutes.

When a device is assigned to a role, any policies applicable to that role will be loaded onto the device and enabled. Thus a temperature sensor would receive policies telling it to read the temperature every <n> minutes but only notify the PDA when the temperature exceeds a threshold. The cardiac sensors may send a 10 second ECG sample every minute to the PDA for anomaly detection, however under some circumstances, it may switch to continuous monitoring with remote transmission via the PDA to a medical centre. We assume that most devices capable of wireless communication are also capable of interpreting simple policies.

Figure 2 shows the interactions between 2 SMCs relating to a home visit from a nurse. The Nurse SMC is discovered and assigned to the Medic Role in the Patient SMC and similarly the Patient SMC will be discovered and assigned to the Patient Role in the Nurse SMC. Each of the SMC PDAs will receive policies relating to its relevant role. The Patient SMC will have authorisation policies permitting members of the medic role to interact directly with its devices. The patient's SMC could also be made a member of the Medic SMC. The nurse could then replace, enable or disable policies in the patient's SMC to modify the medical regime. This type of peer-to-peer relationship is one way of supporting SMC to SMC interactions.

In the following sections we discuss some of the SMC functions in more detail.

4 SMC Architecture Details

4.1 Policy Service

We have had considerable experience of the use of policies as a means of specifying adaptive behaviour in network management and other applications. The use of separate, interpreted policies means they can be easily changed without shutting down or recoding components. Authorisation policies should be enforced on the target components they are protecting as these must make the decision whether to permit or deny access. Requesting a decision from a remote policy interpreter via wireless communication would not be practical in most cases. For example a policy of the form:

auth+ /sensors/temperature $\rightarrow$ /pda.*reportTemp*

would be needed to permit temperature sensors to perform the reportTemp operation on the pda.

Obligation policies are event-condition-access rules and would be associated with a role. For example the following policies would be loaded into a temperature sensor when it is discovered and assigned to the temperature role:

```
// Policies loaded into a Temperature Sensor
5. oblig every (mins (2)) →
      raise local tempEvent (read_temperature())

6. oblig tempEvent (temperature) →
       temperature > MAX_TEMPERATURE ?
           /pda.reportTemp (temperature)
```

Policy 5 tells the sensor to read the temperature and raise an event every 2 minutes and is triggered by a local time event. Policy 6 will report the temperature to the pda when the temperature value is greater than a maximum value. Wireless communication is only used if there is a problem with the temperature and, as a result, this minimises power consumption.

The above policy notation is loosely based on the Ponder toolkit [3][4] developed at Imperial College London. Ponder is being redesigned to be suitable for smaller devices such as PDAs and body sensor nodes (BSNs) developed in the UbiMon project. The latter have very low power 16-bit processors 64KB RAM and 256KB Flash memory, 6 analog channels for sensors and Zigbee radio [1].

Obligation policies can also be used to manage other policies in terms of selecting, enabling and disabling policies [5] to adapt overall behaviour of a SMC to current context e.g. a set of policies for when the patient is sleeping and a different set related to situations when the patient is doing normal activities. Policy 7 is triggered by the sleeping event from the context sensing service, and enables the policies for sleeping and disables those for the awake state, while Policy 8, triggered by the awake event, does the opposite.

```
7. oblig sleepingEvent () →
         /pda/policies/sleeping.enable(),
         /pda/policies/awake.disable()

8. oblig awakeEvent ()→
         /pda/policies/awake.enable(),
         /pda/policies/sleeping.disable()
```

This obviously depends on the ability to detect current activity, which requires fusion of information from multiple sensors and we can make use of other work in the UbiMon project for this.

4.2 Event Bus

We have chosen to implement the event bus as an at-most-once, persistent publish/subscribe delivery service, using a router to distribute events to subscribers. The router is content-based – i.e., a subscriber specifies a filter when it registers, and all published events that match the filter are forwarded to that subscriber. The structure of the event bus is shown in Figure 3.

Publishers do not need to register with the Router. When a publisher sends an event to the Router, it does so synchronously and reliably; this reliable delivery is shown by the request/response arrow. Successful delivery of the event to the Router transfers responsibility for subsequent delivery to the Router. The Router attempts to deliver such an event to each subscriber whose filter matches the event. If it is unable to deliver the event to a particular subscriber due to transient communication failure, it queues the event for redelivery to that subscriber. The router attempts to deliver queued events until it knows that the subscriber is no longer a member of the SMC. Each subscriber is guaranteed to receive all events from a particular publisher in the same order as received by the Router. This is required in case there is a causal relationship between events from a particular publisher. If the Router receives a component-left event, it removes that subscriber's filters (if it had registered for any events), and purges any queued up events for that subscriber.

We do not assume that all communication within the SMC takes place via the event bus. Events are used to trigger policy actions. For example the policy service may use simple broadcast messages and unicast messages for polling individual components. Other components may use remote procedure calls or object invocations depending on what they

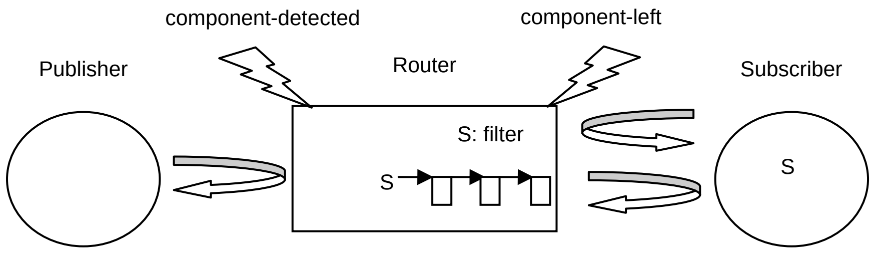

Figure 3 Publish-Subscribe Event Service

support.

A device such as an intelligent sensor may also be considered a very simple SMC as it is capable of interpreting policies. It may support a simplified event service containing only a single subscriber with localised events e.g. from a timer in order to trigger policies at regular intervals.

4.3 Discovery Service

The discovery service is responsible for detecting new devices or other SMCs when they come into communication range. It is responsible for vetting a device for membership by obtaining the device's profile. If a new device passes the vetting procedure, the discovery service generates a component-detected event as indicated in Figure 3. By making the discovery service policy driven, it can easily be adapted to different applications.

This service is also responsible for determining when a device permanently leaves the cell and for generating a corresponding component-left event. Although in a patient SMC, most sensors will not be independently mobile, the nurse SMC in the medic role might go out of the room and hence be out of wireless range for a short time. The discovery service has to distinguish between permanent departures from the cell and temporary loss of communication. Thus the number of failed polls to detect a departure from the cell and the frequency of polling a component is dependent on the SMC application so must be configurable via policies. We have implemented a very simple discovery service which runs on the PDA and broadcasts its identity message (id; type[; extra]) at frequency ω_R. A new device responds to the router identity message with a unicast device identity message. The discovery service can then query the device to obtain a device profile describing the services; it performs some basic vetting of devices, informs the device whether it has been accepted for membership, and if so generates a component-detected event which results in the device being assigned to specific roles depending on its profile and possibly on the credentials it possesses as indicated in Figure 1.

Each member device unicasts its identity message to the discovery service at the frequency ω_D. If the discovery service misses n_D successive messages from a particular device, it concludes that the device has left the SMC permanently, and generates a corresponding component-left event. If the device misses n_R successive discovery service identity broadcasts, it declares that is no longer a member of that cell.

When a device joins an SMC, it will not respond to a discovery service broadcast from another SMC. In the healthcare scenario, a sensor should not decide that it has left the SMC because there is a problem with the discovery service and then join the SMC of the person sitting next to the patient on the bus. One approach is to use pairing buttons on the PDA and device which have to be pressed simultaneously, while the devices are in radio contact, in order to set up an association with a new SMC, as with simple Bluetooth devices. We will investigate other more secure techniques in the Caregrid project [6].

We may also provide access to other discovery services such as UPnP [7] although this does not support any form of adaptation of the discovery service itself.

4.4 Trust, Security and Privacy

A pervasive healthcare system involves complex interactions between many services in many organisations. If an emergency is detected the monitoring service calls an ambulance. The monitoring service needs access to patient cardiac history from the patient's GP and from the hospital where the patient had treatment and so liaises with the emergency services and the hospital to which he will be taken for emergency treatment. The monitoring service also provides anonymised monitoring records for medical research. Trust is a key issue as we want trust-based decisions relating to the interactions between entities – which entity to interact with, what resources should the entity have access to, what information should be released to the entity, how to configure the mechanisms needed to make the interaction secure and how trust levels change over time, based on experience and reputation. Large-scale applications cannot rely on the traditional person in the trust decision loop, but must make use of automated trust decisions.

There are many different aspects of trust in all the above interactions. Will the monitoring service detect actual problems without false alarms? Can the wireless infrastructure used be trusted with respect to confidentiality? If not, can this lack of trust ensure that data is communicated over a secure channel? Can the monitoring service be trusted to pass on monitored information for research while still maintaining patient privacy and can they guarantee that the information will be used only for medical research? Can the patient (if he so wishes) agree to information being sold to insurance companies in exchange for a lower monitoring service charge or insurance premium? In cases of emergency, can we ensure that privacy issues may be over-ruled and the patient's doctor should have access to detailed monitored information? Trust with respect to interactions between organisations (e.g. a hospital using a blood analysis service) will change over time based on experience, recommendations, or reputations [8]. There is a need to collect this evidence for use in making decisions based on trust, for example in health workflow systems to aid medical procedures or patient care, where the entities participating in the workflow, change dynamically because of workload, availability etc and may have varying levels of trust between them. Trust may also depend on current context, particularly for mobile applications. Privacy – an individual's right to control the collection and use of personal information plays a crucial role in building trust, particularly in healthcare applications.

Most trust-based systems rely on the ability to validate credentials, which implies access to certification authorities and credential revocation lists, but internet access cannot be guaranteed from and SMC. Other techniques are needed for using recommendations or assertions of the trustworthiness from other entities within an SMC as described in [9].

Trust and security were explicitly excluded from the Amuse project, but will be addressed in the EPSRC funded Caregrid project [6].

5 SMC Simulator

To explore SMCs, we are implementing several prototype systems. The logistics of building such prototypes dictates that while they will be functional, we will not be able to significantly explore the scalability aspects of such systems, nor will we be able to study peering interactions between large numbers of SMCs. In order to explore these scale issues, we have implemented a discrete event simulator for SMCs [14]. The simulator will provide insight into the operation of individual SMCs and the interaction between internal components, allowing an enhanced generic SMC architecture to be designed.

The simulator will enable the accurate simulation of management traffic of an SMC in an attempt to optimise component functionality and component interaction. Measurements such as the average delay from the time an event is signaled until interested components receive it will becomes apparent as will the time it takes the event service to propagate events.

We will further be able to test different discovery mechanisms, inferencing engines and event buses, accurately visualising the effect these have on the overall functionality of an SMC. It will also be possible to model the federation, layering and composition of SMCs into larger structures.

An initial version of the simulator has been completed, and is documented in [14]. Additional required simulator functionality will be added throughout the project.

6 Related Work

IBM has been the prime mover towards autonomic computing [10] and HP is also addressing similar issues in on-demand Utility Data Centres [11]. However most of the industrial work focuses on large clusters and web servers whereas we are concentrating on pervasive computing which is potentially more dynamic due to the mobility of components.

The Universal Plug and Play (UPnP) Architecture supports resource discovery and configuration of consumer devices (TV, video recorder, air conditioning etc.) which communicate via wireless within a home or office [7]. Although they concentrate on device configuration rather than configuration of software within nodes, some of the protocols and XML service specifications can be adapted for our purposes. UPnP currently focuses primarily on self-configuration and does not support the adaptability required for healing, optimising or protecting.

There are many publish-subscribe event services such as Elvin [11], XML-blaster [12], and Sienna [13], which we have used in test systems; unfortunately, none of these routers are designed to run on small devices such as body sensor nodes (BSNs) and PDAs.

7 Status and Future Work

We are currently building an implementation of the SMC architecture. The prototype consists of a set of body sensor nodes with Zigbee wireless capability developed in the UbiMon project that communicate by low-power radio with a PDA that hosts the management components and has wireless LAN or GPRS communications. There are a number of issues still to be resolved, such as making sure the protocols we develop optimise the use of battery power; how to make sure a device is ‘owned’ by a particular SMC and cannot be taken over by another SMC; how SMCs can interact at a level of abstraction higher than implementable policies etc. Three related projects are starting which will investigate the privacy trust and security issues; the applicability of the concepts to SMCs consisting of groups of unmanned vehicles which need to cooperate to achieve an overall goal; the applicability to micro-miniaturised and implantable sensors; how to define, gather, combine context information for triggering and constraining policies.

Acknowledgements

The authors wish to thank the UK Engineering and Physical Sciences Research Council for their support of this research through grants GR/S68040/01 and GR/S68033/01.

References

[1] DTI funded UbiMon Project http://www.ubimon.org

[2] DTI funded Care in the Community Project http://www.dticareinthecommunity.com

[3] Damianou N., N. Dulay, E. Lupu, M Sloman, The Ponder Specification Language Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 18-39

[4] Damianou N., N. Dulay, E. Lupu, M. Sloman, T. Tonouchi, Tools for Domain-based Policy Management of Distributed Systems, *IEEE/IFIP Network Operations and Management Symposium (NOMS2002)*, Florence, Italy, 15-19 April, 2002

[5] Lymberopoulos L., E. Lupu and M. Sloman. An Adaptive Policy Based Framework for Network Services Management, *Plenum Press Journal of Network and Systems Management,* Special Issue on Policy Based Management, 11: 3 Sep. 2003, pp277-303

[6] EPSRC CareGrid Project http://www.doc.ic.ac.uk/%7End/projects/CareGrid.html

[7] Universal Plug and Play Device Architecture. http://www.upnp.org/resources/documents.asp

[8] Abdul-Rahman A. and Hailes S.: Supporting Trust in Virtual Communities, Proc. of 33rd Annual Hawaii Intl. Conf. on System Sciences, Hawaii, Jan. 2000, Vol 1, 9pp

[9] S.L Keoh, E. Lupu, M. Sloman, PEACE : A Policy-based Establishment of Ad-hoc Communities, IEEE Annual Computer Security Applications Conference (ACSAC 2004), Tucson, Arizona, USA, Dec. 2004, pp 386-395

[10] Autonomic Computing Special Issue, IBM Systems Journal, Vol. 42, No 1, 2003.

[11] HP Utility Data Center: Enabling Enhanced Datacenter Agility, http://www.hp.com/large/globalsolutions/ae/pdfs/udc_enabling.pdf, May 2003

[12] Elvin http://www.mantara.com/

[13] xmlBlaster http://www.xmlblaster.org/

[14] Siena Wide Area Event Notification Content Based Routing http://serl.cs.colorado.edu/~serl/dot/siena.html

[15] Heeps, S. and O. Sharma, “Self-Managed Cell Simulator”, University of Glasgow Technical Report, February 2005.