Policy Issues for Pervasive Systems

Morris Sloman
Department of Computing, Imperial College London, London SW7 2AZ.
m.sloman@imperial.ac.uk
http://www.doc.ic.ac.uk/~mss/

Pervasive or ubiquitous computing systems consist of large numbers of 'invisible' computers embedded into the environment which may interact with mobile users or form intelligent networks for sensing environmental conditions. Users will experience this world through a wide variety of devices, some they will wear (e.g medical monitoring systems), some they will carry (e.g. personal communicators that integrate mobile phones and PDAs), and some that are implanted in the vehicles they use (e.g car information systems). This heterogeneous collection of devices will interact with intelligent sensors and actuators embedded in our homes, offices, transportation systems to form an intelligent pervasive environment which aids normal activities related to work, education, entertainment or healthcare.

The promise of such ubiquitous computing environments will not be realised unless these systems can effectively "disappear"; and for this they need to become autonomous by managing their own evolution and configuration changes without explicit user or administrator action. Developing the architecture, tools and techniques which permit these environments to become self-managing is therefore essential. Self-management must apply at all levels: for individual devices, for simple body-area networks, for embedded devices within the home or the work environment, as well as for large-scale network infrastructures and inter-organisational applications.

We advocate the concept of a self-managed cell (SMC) as the basic architectural pattern for implementing self-management at both local and integrated levels. A self-managed cell consists of a set of hardware or software components which form an administrative domain that is able to function autonomously and thus is capable of self-management. A SMC could represent the resources available in a PDA, a body area network of physiological sensors and controllers, as well as the application components relating to a set of collaborating partners forming a virtual (e-Health) organisation spanning multiple countries. In each case SMCs must be automatically configured with the required management services, appropriate to the scale and environment of the cell. These services interact with each other through asynchronous events exchanged over an event bus (see Figure 1). As a minimum, a SMC should contain functionality for measurement and event correlation and support for policy-based control, where policies should specify which adaptation should occur in response to changes of state in the managed resource or changes of context in the environment. In essence, a SMC is a "closed-loop" system where changes of state in the resources trigger adaptation which in turn affects the state of the system. In a ubiquitous environment, the SMC would also typically include management components which provide contextual information and service discovery components (see Figure 1).

Note that the list of management services which are shown in Figure 1 is by no means exhaustive. As the SMC closed-loop pattern is applied to larger structures such as organisational networks, other management services such as accounting, resource planning, optimisation and analysis would become essential. In fact, one of the most important challenges that needs to be addressed is how to define a SMC so that it is extensible, and can be specialised to particular environments such as personal area networks, appliance networks, as well as large scale distributed applications.

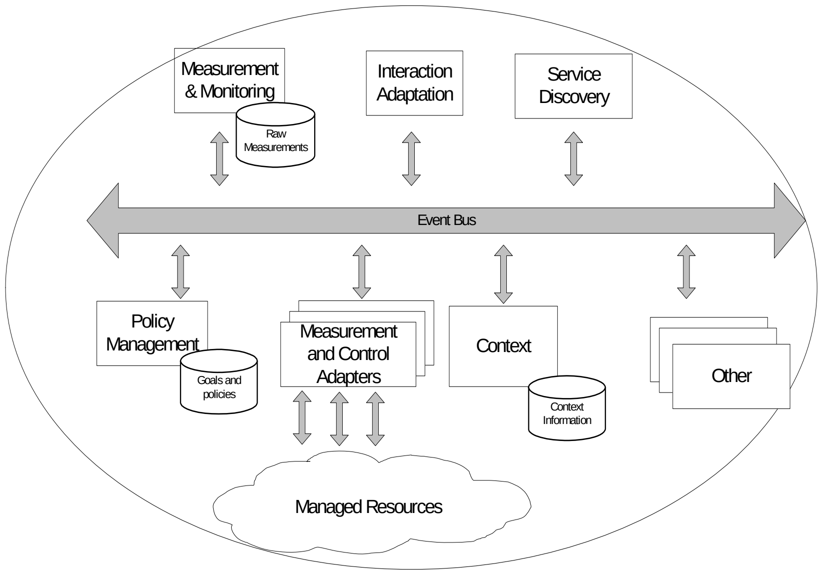

Figure 1 A Self-Managed Cell (SMC)

Although self-managed cells provide the management capability for supporting configuration and adaptation at each level of abstraction, there is a need to support management across multiple cells forming a larger application and/or at multiple levels of abstraction for composed services. We have identified the following forms of management interaction:

- *Composition,* where composed SMCs form a single administration domain and nested SMCs are not visible to external SMCs i.e. any management interaction is via the encapsulating SMC.
- *Federated* to support peer-to-peer interactions between SMCs in order to collaborate and integrate to provide a service e.g. communication subnets or a team of health workers collaborating in the care of a patient. The management relationships between federated SMCs are often transient, but can be long-lived.
- *Layered* to support interactions between management services that require lower-layer management services, for example a medical monitoring service that requires a wireless communication service as well as a storage area network for storing large quantities of monitored data. Note that a management service within a SMC will typically interact with multiple independent management services both above

it (that use it) and below it (that it requires). The concept of hierarchical layering of services is commonly used in communication protocols and web services.

Here, as well, some of the classical network management techniques can be applied. For example, service level agreements (SLAs) can be used to formalise interactions between peer SMCs – although further work is required, as SLAs traditionally focus on network quality of service parameters such as delay or bandwidth and do not address issues.

One of the key issues for autonomic (self-organising) systems is how to define the strategy used for self-management. Generally some form of policy is advocated but the level of abstraction at which policy is defined is still an issue. The simplest approach is to define management policy in the form of event-condition-action rules [3]. The policy is interpreted and so can be modified dynamically, however it is a fairly low level of abstraction. There is some work on specifying policy in the form of utility functions or high level goals and using AI planning techniques [4] or policy refinement techniques [2] to transform the abstract goals into ECA rules. The use of a utility function as the means of specifying strategy requires a detailed model of behaviour of the system in order to determine what actions to take to achieve the required utility. This can be difficult, if not impossible, to determine for complex rapidly changing pervasive system. The planning and policy refinement systems are comparatively complex and unlikely to be suitable for the simple devices found in pervasive systems.

Policy is not just ECA rules as there is a need to define authorisation policies [3] which define what resources will be shared or services offered when collaborating and interacting with other members of an SMC. The approach we advocate is to define potential 'members' or collaboration partners in the form of roles and then assign discovered entities into the relevant role according to a role assignment policy [5]. However there are problems of validating credentials when some devices may not have access to the internet to check attribute certificates or revocation lists. We make use of a trust based system where the members of the cell collaborate to accept new members by issuing trust assertions based on a-priori knowledge of a new member or cached certificates.

The presence of multiple policies relating to management and authorisation, potentially specified by different people can lead to conflicts and inconsistencies. We use Event Calculus in conjunction with abductive reasoning techniques to detect the existence of potential conflicts in partial specification and generate explanations for the conditions under which the conflicts arise [1].

This talk will discuss some of the above policy issues related to policy for ubiquitous computing.

1. Bandara A.K., E.C. Lupu, and A. Russo, "Using Event Calculus to Formalise Policy Specification and Analysis," presented at 4th IEEE Workshop on Policies for Networks and Distributed Systems (Policy 2003), Lake Como, Italy, 2003, pp 26-39.

2. Bandara AK, Lupu E.C., Moffet J, Russo A. A Goal-based Approach to Policy Refinement, *Proceedings 5th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2004)* IBM TJ Watson Research Centre, New York, USA, June 2004, pp. 229-23
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. Kephart, J.O., Walsh, W.E. An Artificial Intelligence Perspective on Autonomic Computing Policies, *Proceedings 5th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2004)* IBM TJ Watson Research Centre, New York, USA, June 2004, pp. 3-12
5. Keoh S.L, Lupu E, Sloman M, *PEACE* : A Policy-based Establishment of Ad-hoc Communities, *Annual Computer Security Applications Conference (ACSAC 2004),* Tucson, Arizona, USA, Dec. 2004