

# Autonomic computing for pervasive ICT — a whole-system perspective

M Shackleton, F Saffre, R Tateson, E Bonsma and C Roadknight

---

*It is unlikely that we can expect to apply traditional centralised management approaches to large-scale pervasive computing scenarios. Approaches that require manual intervention for system management will similarly not be sustainable in the context of future deployments considering their scale and their dynamic (or mobile) nature. This situation motivates the need to apply 'autonomic' techniques to system management, where the behaviour of whole systems results from the inherent properties that have been engineered in, i.e. such systems need to be adaptive, reliable and self-managing at the 'whole system' level.*

*In this paper we outline a number of design principles that can be applied to create systems that are autonomic in their operation. We focus particularly on generating (and analysing) global system behaviour that arises from the carefully designed interactions of the system components, rather than on the individual behaviour of the components themselves. The design heuristics that we derive (which are often nature-inspired) are illustrated in the context of a number of examples that show how the use of the appropriate principles can lead to the inherent global behaviours that we desire. The result is self-managing, self-repairing systems that can be easily deployed, thus reducing total cost of ownership and increasing overall system reliability.*

---

## 1. Introduction

Pervasive ICT [1] heralds a world full of vast numbers of devices and software entities that are able to communicate with one another. For devices, the communication will typically be via wireless networking technologies such as Wi-Fi, Bluetooth or 3G. A key challenge to realising the pervasive ICT vision, or at least to make it a truly useful vision, is the development of an associated set of technologies that will allow these underlying components to be assembled in real time, to provide useful applications and services. Initial progress towards this goal is proceeding under several initiatives such as Web Services, GRID computing, and peer-to-peer (P2P) computing. These areas in fact have many of the same fundamental underlying concerns in common.

Even if we leave pervasive ICT to one side for a moment, there is already a recognition that existing ICT systems are growing in complexity to such an extent that they are becoming unmanageable [2]. This complexity stems from the sheer scale of current and envisaged ICT deployments, the heterogeneity of their infrastructural components, and the unanticipated interactions between these components that may lead to failures or sub-optimal system behaviour. The cost of the hardware itself continues to fall, but the cost of

trouble-shooting this complexity and fixing the associated problems is increasingly dominating the total cost of ownership (TCO) of large ICT deployments. All of the major IT companies have programmes seeking to address these underlying issues, but perhaps it is IBM that has enunciated the issues and its approach to addressing them most clearly by launching its 'Autonomic Computing' initiative [3]. This initiative is one example of a 'nature-inspired' approach in that it draws inspiration from the human autonomic nervous system where many functions, such as breathing or heart rate, are regulated by the system itself without conscious effort. By analogy, IBM is seeking to create ICT systems that are more 'autonomic' (self-managing) in the following ways:

- self-configuring — adaptation to IT system changes such as new nodes becoming available, or going off-line,
- self-optimising — tuning resources and load balancing,
- self-protecting — guard against damage from attacks or failures,
- self-healing — recovery from, or work around, failed components.

The manner in which IBM is seeking to achieve its goal of dealing with system complexity and making systems which are increasingly autonomic, and thus significantly reducing the TCO of ICT deployments, is described in its autonomic computing blueprint [4], roadmap [5], and associated documents [6]. We agree that these are important challenges for research in ICT today, and would go further to assert that the vision of pervasive ICT could not be realised at the scales envisaged without the development of these associated new technologies. In addition, such self-managing solutions will play an important role in making the pervasive environment ‘invisible’ so that users need not become full-time system administrators to their pervasive home environment and devices.

This paper presents four example systems that between them exhibit behaviour and associated engineering approaches that cover each of the self-managing characteristics listed above. We introduce these example systems for two reasons.

Firstly, they tackle problems with real-world relevance that begin to highlight design heuristics/principles that have proved useful to realising autonomic and adaptive solutions. We expect that this class of approach to designing autonomic solutions will similarly prove useful in tackling other problems and will, after further refinement, become part of mainstream ICT system design methodologies in the future.

Secondly, the example systems highlight the importance of certain approaches to design that are not yet very well addressed by the existing autonomic computing initiative (or associated research). However, we expect they will prove to be important to realising future robust, large-scale and self-managing ICT deployments. In particular, these approaches to system design seek to tackle the relative importance of the interactions between components on the overall whole-system (or global) behaviour. This is illustrated in Fig 1.

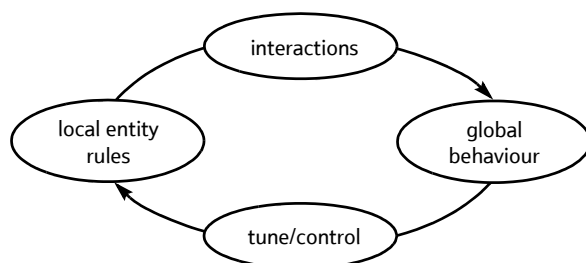


Fig 1 A view of how components within a complex system interact to produce overall global behaviour. The diagram also shows how the design can be iteratively refined, by observing global behaviour and tuning local rules accordingly, with ‘complex systems analysis’ playing a useful role in this cycle.

## 2. Illustrative example systems

The four following examples describe systems that have successfully tackled real-world problems in the diverse application domains of mobile networks, network security, P2P information management, and adaptive service provision. Referring back to the IBM autonomic classifications, these examples address problems involving, respectively, self-optimising resource allocation, self-protecting security management, a self-configuring information space, and self-optimising (load-balanced) service provision. Several of the examples also exhibit self-healing behaviour, in that they can cope robustly with individual components/nodes that fail while maintaining appropriate overall system-level behaviour.

### 2.1 ‘Flyphones’ — channel (resource) allocation

The allocation of channels in a mobile telephone network is one example of a range of problems involving allocating resources in geographically or logically dispersed systems. The operator of the mobile network must decide which base-stations will be permitted to use which channels. This choice is constrained by the conflicting desires to provide each base-station with sufficient channels to meet peak call demand, while avoiding situations in which the channels in use at neighbouring base-stations cause interference and hence low quality of calls.

Channel allocation, in common with most problems of this type, has traditionally been solved by centralised optimisation. Information about the network, gathered over some representative period of time (e.g. one month), is fed in to an optimisation algorithm such as a simulated annealer. Simulated annealing, and related techniques, rely on the idea that an example solution to the problem can be assigned a ‘score’ based on the predicted performance of the entire network if this solution was adopted. By producing a series of subtly different solutions and comparing their scores, such algorithms are able to find their way to better solutions.

However, this approach relies on the ability to accurately model the network such that every proposed solution can be given a score. In situations where accurate models are not available, either because information is unavailable or out-dated, and particularly if the network is changing rapidly such that good solutions themselves quickly become useless, it is desirable to have a dynamic channel allocation strategy which can provide good performance without global knowledge.

We have produced such an algorithm by drawing inspiration from nature. The cells in developing animals are able to produce a highly detailed and accurate macro-structure (the adult form of the animal) without

any global knowledge. The cells rely on interactions with their immediate and mid-range neighbours to self-organise. One example is bristle formation in the fruit fly *Drosophila melanogaster*. Many cells in the developing larva have the ability to form bristles, and they send inhibitory signals to their neighbours indicating that those neighbours (who may be making such signals of their own) should not form bristles. This mutually inhibitory ‘conversation’ continues for a few hours, by the end of which most cells have abandoned the ability to make bristles, leaving a few ‘winners’ to go ahead and produce these structures in the adult fly.

To produce a channel allocation algorithm inspired by this process requires two steps. Firstly, it is necessary to sacrifice central control and allow the base-stations the same level of autonomy enjoyed by the fruit fly cells. The base-stations can decide for themselves which channels they will use. Then we provide the base-stations with the ‘mutual inhibition’ logic (local rules) for interacting with their neighbours (see Fig 2). Put simply, each base-station must send signals to its neighbours attempting to stop them from using its ‘favourite’ channels, and it must respond to such signals from its neighbours by reducing its ‘preference’ for their favourite channels. Sometimes there will be a ‘clash’ whereby two base-stations both want a particular channel, and then just as in the fruit fly, this will be resolved with one base-station emerging victorious and the other relinquishing that channel.

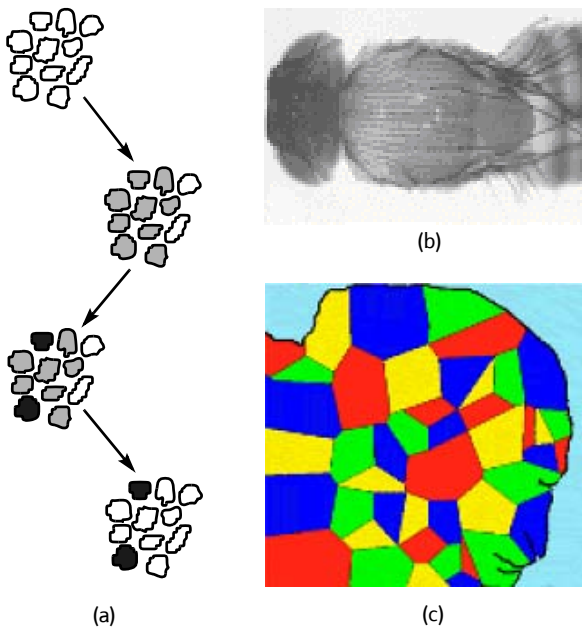


Fig 2 (a) The mutual inhibition process which limits the number of cells adopting the neural ‘bristle-making’ fate. Many have the potential (grey) but only a few realise that potential (black) while inhibiting their neighbours (white). (b) The pattern of bristles that results in the adult fly. (c) A map of East Anglia in the UK, with the coverage areas of base-stations in a mobile telephone network shaded as if there were only four frequencies to allocate.

The algorithm has been tested in network simulations of both civilian and military wireless communications networks spanning the range from fairly simple to rather complex scenarios [7,8,9]. It successfully self-configures to provide good solutions to the global problem. More importantly it brings with it the ability to continue to function well even in a network which is losing or gaining nodes, where nodes themselves may move and hence encounter new neighbours, and where accurate up-to-date information about the overall network is not available. In this sense it is also self-healing. The algorithm also evades scaling problems since it relies on the elements of the network to perform their own local decision-making — hence as the network gets bigger, its processing power increases exactly in step with this growth.

## 2.2 ‘ADDICT’ — adaptive response to threat

Network security in general and intrusion detection (IDS) and response in particular provide another striking example of a context where the cost and practical limitations of centralised management become more evident by the day. The combination of ever-increasing network speed/capacity with the rapid evolution of new, more efficient ‘cyber-threats’ (e.g. ultra-fast worms) has created a very volatile situation. Even if methods for monitoring network activity are available, it is a fact of life that data cannot be analysed, tactical decisions made and defensive measures implemented in time to contain a developing threat if all these processes are taking place centrally.

Of course, pervasive computing, in the form of enhanced mobility, portability and diversity of interacting devices, is contributing to make this situation worse. Seamless connectivity and ubiquitous access effectively signal the end of any permanent distinction between secure and insecure network environments, a phenomenon that has been adequately named ‘the disappearing perimeter’ [10]. In practice, unless a third way can be found, we are facing a bleak alternative between denying services by default (which goes against the very principles of pervasive ICT and somewhat defeats its purpose) and leaving our networks wide open to attack.

Concretely, what is needed is a method to bridge the gap between a host-based (HIDS) and a network-based (NIDS) intrusion detection system. The former typically scales well because every device is in charge of monitoring its immediate neighbourhood (and so processing power grows linearly with network size). On the other hand, it is incapable of recognising suspicious global activity patterns, because scattered data does not allow for the distinction between ‘security noise’ (e.g. the odd probing of IP addresses) and a concerted attack by a malicious entity. The latter is an archetypal

example of the shortcomings of centralised management — if infinite processing power was available, NIDS would in theory be capable of immediately picking up subtle correlations between events happening throughout the network and identify a serious threat. However, in practice, this ideal case is obviously never realised and so network-based intrusion detection is usually conducted off-line, which turns it into a sophisticated forensic tool rather than a real-time protection system.

We have proposed that a biologically inspired form of inhibitory signalling between hosts could be used to overcome both difficulties. In 'ADDICT' [11], individual devices compute an 'alert level' on the basis of locally detectable network activity (equivalent to HIDS), and then exchange beacon signals with other nodes, which are effectively a digest of security-relevant information (i.e. the alert level, but also identity, internal state, etc). The collected beacons are then used by every member to update its own alert level  $x$  as per equation (1) below:

$$\frac{dx}{dt} = \frac{x(1-x)}{N} \left( N - n + \alpha \sum_{i=1}^n x_i \right) - \beta x \quad \dots (1)$$

where  $N$  is the number of perceived devices (sources of network traffic),  $n \leq N$  is the number of identifiable/trusted devices among them, and  $\alpha$  and  $\beta$  are tunable parameters. This creates a feedback loop allowing the community of peers to rapidly converge toward a state reflective of the globally perceived threat level (similar to NIDS).

Once the alert level is mapped on to predefined security stances (probably involving use of a personal firewall), ADDICT, which has been called 'the equivalent of Seti@Home [12] for network security', should be capable of rapid adaptation to changing conditions and real-time redrawing of the trusted domain boundary. This dynamic perimeter defence can accurately be described as an enabler for self-configuring, self-protecting and self-healing pervasive networks as it could allow for immediate reaction to security breaches as they are forming (e.g. aggressive scanning by a rogue device) and spontaneous containment of the corresponding threat through isolation of all sources of suspicious activity.

### 2.3 'SWAN' — self-organised information management

The peer-to-peer (P2P) paradigm fits autonomic computing very well. P2P infrastructures contain many nodes, and lack centralised control. There can be large variations in the capabilities of the peer nodes. They are typically unreliable, and nodes can join and leave the system at any time. Despite this, P2P applications can

be built that function reliably, automatically adapt to changes, optimise their configuration, and can cope with node failure and malicious attacks. This is made possible by various P2P technologies. One such technology is the small world adaptive network (SWAN) [13]. It is a distributed look-up system for P2P networks that can be used to find out the address of a node, given the key under which it is published. Nodes here can be anything that can be accessed through an underlying communications network, including computers, documents, services and users. An important property of SWAN is that it is fully decentralised. This is done to make the system robust to failure and to ensure that the load is shared fairly across all computing nodes that contribute and participate in the look-up system.

SWAN is able to provide look-up functionality without resorting to centralised tables by letting nodes self-organise into a virtual network. Each node simply maintains a limited number of links to other nodes in the network. These links are used for handling look-up queries. For each incoming query, a node examines its links to find the node with a key that is closest to that of the target query. It then forwards the query to this neighbouring node, unless its own key is actually closer to the query. In the latter case, the receiving node replies with its own address. So each node uses only simple, local rules for handling incoming queries.

Whether or not queries can be successfully answered this way, and the number of messages that are on average required per query, depends on the links that the nodes maintain. SWAN is engineered so that nodes self-organise into a network that meets specific properties. When these properties are met, queries are guaranteed to succeed and the number of messages that are required scales well with the total number of nodes.

More specifically, nodes self-organise into a 'small world network' [14]. Each node maintains a limited number of short-range links to nodes with different but similar keys. These links ensure that all nodes can be found. If, however, only these links were used, the number of messages that is needed does not scale well. Therefore, nodes also maintain a fixed number of long-range links to nodes with keys that differ more. Although these links are randomly chosen, the random process is designed such that the long-range links meet the distribution of lengths that is recommended by Kleinberg, which he mathematically proves to be optimal [15, 16]. This way, the number of messages required to find a node is proportional to  $(\log N)^2$ , where  $N$  is the total number of nodes.

The SWAN system illustrates that a good way to build a decentralised system is to base it on a solid

theoretical foundation. In this case Kleinberg's theoretical work showed what the global properties of the small world network needed to be for nodes to effectively route queries using only local knowledge. Obviously, engineering the system may still involve tackling difficult issues. In the case of SWAN, for instance, the question of how nodes can self-organise into a network with the desired global properties, using only local knowledge and interactions, still has to be solved.

#### 2.4 'Bacterial plasmids' — adaptive service provision on an active network

Data networks are continually growing and changing. Management and control of these networks will also need to change if they are to deliver an acceptable quality of service. External human control becomes untenable in most envisaged scenarios, especially if the network becomes more 'active'. Positioning the right active software in the right place at the right time is a combinatorially explosive problem.

We have proposed and tested a scalable solution to adaptive management of active service networks [17, 18] which suggests that a bacterium-inspired software distribution algorithm is well suited to the task. The approach treats mobile software and associated execution constraints as bacterial RNA (ribonucleic acid) and the execution environment in which it is run as the bacteria itself. The 'RNA' decides how suited the bacterium is to its current location. So if the code is profitable, sections of the RNA (plasmids) spread around the network in search of further profitable locations. Alternatively, if the code is unused or unprofitable its fitness declines until it is replaced or modified by RNA from a more successful organism. Figure 3 shows a possible future environment where active code inhabits a network of dynamic proxy servers. Requests for the services that the software provides are handled if the code is present and the subjective criteria (busyness, queue lengths, cost, price, etc) are matched. The combination of code present on the device and the subjective criteria for use of this code amounts to the RNA of the node.

Mimicking the unmatched haste of bacterial evolution gives the network the required adaptability, while its distributed nature is inherently more robust than a centralised management approach. Genetic diversity is created in at least two ways, namely mutation and plasmid migration. Mutation involves the random alteration of just one value in a single rule. Plasmid migration involves genes from healthy individuals being shed or replicated into the environment and subsequently being absorbed into the genetic material of less healthy individuals. This intra-

generation (bacterial) evolution adds a level of short term adaptation over the longer term inter-generation (Darwinian) evolution.

As load balancing and the distribution of new software are implicit parts of the reproductive and evolutionary nature of the algorithm, it is no surprise that these functions are performed so readily [17]. More complex functionality, such as allowing some varied quality of service, some payment-based security and the ability to handle realistic traffic streams, were also demonstrated [18].

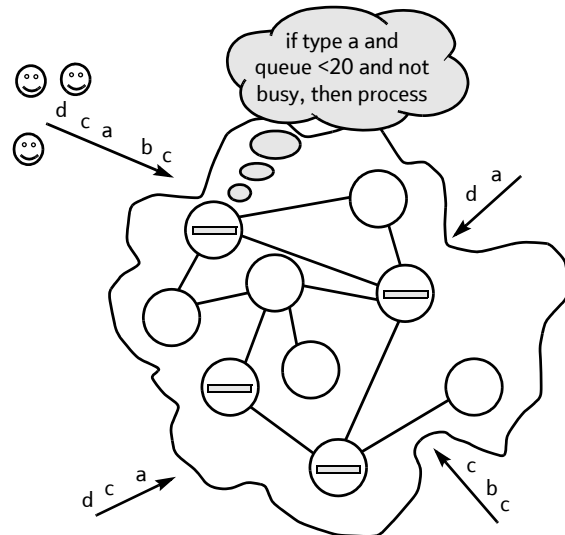


Fig 3 Future 'active' network model. Dynamic proxy server nodes with active code have a bar, those without are empty. Incoming requests (a,b,c,d) are routed to the nearest live node by an underlying transport network.

While external management is not required, it is supported in the guise of injectable policies. Active software can be manually placed on the network, with associated policies for its use; its adoption and proliferation will depend on user behaviour and this initial placement. The configuration can also be manually modified by injecting new fitness functions, e.g. instating a fitness function that gives no reward for running software would soon purge the network. This combination of 'hands-off' management with simple methods to enable intervention is a key requirement for the adoption of autonomic solutions, as it supports policy-driven management.

### 3. Discussion of example systems

The example systems discussed above cover a wide range of application and problem domains. However, each example exhibits one or more of the 'autonomic' self-managing characteristics — self-configuring, self-optimising, self-protecting and/or self-healing. We will now consider certain common features that are highlighted by the examples that may prove useful as

design heuristics for the creation of future autonomic ICT systems. It is probably worth noting that autonomic computing defines a series of maturity levels [19] — basic, managed, predictive, adaptive, autonomic. The examples we have described should best be considered as ‘adaptive’ (i.e. ‘takes action itself based on the situation’) or possibly fully ‘autonomic’ (i.e. policy-driven system activities, such as resource allocation).

Clearly the autonomic computing initiative is biologically inspired in origin, in that it explicitly refers to (and aspires to mimic) the characteristics of the human autonomic nervous system. This natural system regulates many bodily functions in humans, such as heart rate. It is worth noting that three out of the four examples given earlier are themselves ‘nature-inspired systems’ (NISs). In fact, we have found nature to be a rich source of design principles for robust and decentralised architectures [20]. By looking at analogous natural systems that exhibit desirable properties, we can often gain insights into how artificial systems may be constructed that exhibit similar properties. For example, the mammalian immune system has been used as inspiration to devise novel self-protecting mechanisms for computational systems [21]. However, it is also possible to use analogous design heuristics regardless of a specific natural analogy, as was done for the design of the ‘SWAN’ information management example described earlier.

So what design heuristics might we draw from the examples we have given? We suggest that the following are some key principles:

- local rules — wherever possible use local rules and decision making to achieve overall behaviour,
- interactions — by combining local decision-making with carefully crafted interactions between neighbourhood nodes/entities, the desired global behaviour can often be achieved,
- positive and negative feedback — in the same way that biological systems make extensive use of feedback to control processes and achieve robust design of structure and behaviour, similar principles can be used in artificial systems, e.g. Flyphones and ADDICT use inhibitory signalling,
- decentralised solutions — often a given problem is in essence a decentralised problem (cf Flyphones), where a decentralised solution may be well matched, since, in addition, nodes in a decentralised system often ‘bring their own resources’ which can help provide a scalable solution,

- engineered-in behaviour versus explicit external control — where possible, it is preferable to embody some management within the system itself, as policy-based management is still appropriate and possible via tuning parameters and via the system’s in-built adaptability,
- complex systems — the analysis tools of complex systems research are likely to prove increasingly important, e.g. this approach helped with the design of the SWAN system.

These design heuristics are already useful for creating autonomic solutions, as can be seen from the example systems discussed earlier. We can speculate that it may be possible to begin to formalise such heuristics as design patterns [22] for pervasive ICT systems in the future.

In fact, some well-known existing systems make use of a number of these principles, whether implicitly or explicitly. For example, peer-to-peer systems (such as KaZaA) create a decentralised architecture that makes use of local rules and neighbourhood interactions to create an autonomic system that is clearly self-healing and self-protecting.

The IBM Autonomic Computing initiative has a strong focus on managing the components in a system. These ‘managed elements’ are controlled through associated sensors and effectors [23]. We believe that currently there is insufficient focus on the nature of interactions between the components. While ‘autonomic manager collaboration’ is explicitly mentioned in the IBM autonomic computing blueprint [4], it simply says that the ‘... autonomic managers can communicate with each other in both peer-to-peer and hierarchical arrangements’. It notes that the ‘... autonomic managers for the various contexts will need to cooperate’ using a ‘matrix management protocol’. In a pervasive ICT context with many interacting components, it is unclear how this management will be effectively resolved.

The research field of complex systems deals specifically with systems of many interacting components, and helps explain how overall system behaviour arises as a result of these interactions. It offers the potential to allow the behaviour of systems to be analysed, tuned, and bounded appropriately via analytical, modelling and simulation-based approaches and tools. It also highlights that in many cases the interactions are a greater determinant of global system behaviour than the individual components themselves. Complex systems theory can help assess and quantify the robustness and dependability of solutions, such as specific network architectures [11]. It can also suggest

how local rules may be created to achieve the desired global behaviour (e.g. this approach was used in the 'SWAN' example above to engineer an efficient solution.) We also note that some others have observed that complex systems theory has a role to play in realising autonomic systems [24].

Just as for the design of current ICT systems, designers of autonomic solutions will have to ask themselves whether there may be some unintended side-effect or interaction between sub-systems or components that may lead to failure conditions or inefficiencies. For these reasons complex systems theory and modelling tools are likely to prove important in engineering robust and adaptive solutions in a pervasive ICT context that by definition includes many interacting components. The next section discusses this in more detail.

#### 4. The need for 'complex systems' theory and modelling

Traditional engineering starts by specifying desirable system-wide characteristics and then designs/selects individual components under the assumption that the whole is only the sum of its parts. In the domain of pervasive computing this approach is unlikely to deliver viable solutions due to the scale and complexity of the overall system. The mobility of participating devices and diversity of available services adds an additional dimension to the already arduous problem of managing large distributed systems. However, despite being aware of this difficulty, many technologists seem reluctant to cross the cultural barrier between a proven and immensely successful paradigm (inherited from the industrial revolution) and the 'new' science of complexity, which is less well understood by engineers.

Complexity science provides powerful methods for dealing with probabilistic predictability and describing systems comprised of individually unpredictable elements in a rigorous and useful way. Over the last three decades, it has been extensively demonstrated that variability in the individual response of its constituents does not necessarily translate into the frequency distribution of a system's states exhibiting a similar amount of 'noise'. On the contrary, the huge number of interactions and the presence of intricate feedback loops often mean that the system as a whole can only exist in a limited number of configurations, despite the largely random behaviour of individual units. The science of complexity mainly consists of identifying these configurations, determining their probability of occurrence, and understanding/characterising transitions between them and trajectories leading to them (e.g. bifurcation).

The sheer size of a large network comprised of many thousands of components means that the state of a pervasive computing environment will virtually always be the result of an unforeseeable combination of many events, and so can only be described probabilistically. While there may be an increased recognition of this situation, there is a poor awareness of the methods capable of dealing with it. The heterogeneity of the underlying infrastructure (in terms of purpose, capability, and ownership) precludes a centrally imposed set of rules defining the function and privileges of every participant. Instead, we must find ways to engineer autonomic principles, like self-configuration, into individual elements and their interactions, so as to allow them to deal with unexpected situations, requests, combinations of events, etc.

Complex systems theory and modelling can and must help us understand which macroscopic behaviour is more or less likely to emerge from the many interactions between heterogeneous devices. The real challenge is not to cope with microscopic unpredictability — the conceptual tools required to handle its macroscopic effects are readily available. The difficulty resides in identifying and weighting the factors involved, so that the purpose of fine-tuning the local rules is not defeated by the presence of 'hidden variables' capable of pushing the entire system into an unexpected/undesirable state.

#### 5. Conclusions

The existing trends of ICT systems are already considered to be unsustainable in the long run [2]. This is because the complexity and scale of ICT deployments are outstripping the capabilities of our existing design tools and manual management practices. The impact is experienced in the form of increasing total cost of ownership (TCO) of large ICT deployments, as well as reliability issues and application down-time. All of the major IT companies have programmes seeking radical solutions to these problems, including IBM's Autonomic Computing research initiative [6]. As we enter the world of pervasive ICT, the envisaged scale and complexity makes 'autonomic' self-managing solutions not only desirable, but essential.

This paper makes explicit a number of design heuristics that are particularly useful for implementing 'autonomic solutions'. These principles have been illustrated in the context of four example systems that address specific real-world problems in ICT. Note that three of the four systems are directly nature-inspired, while the remaining example uses similar design principles in a 'fully engineered' solution. The use of the appropriate principles leads to efficient solutions with self-managing properties and real-time adaptive and resilient behaviour. By implication, this is helping to

address the issue of reducing the TCO of ICT deployments, and coping with system complexity.

In addition to presenting design principles that can be used right now to engineer-in autonomic behaviour, we also point towards the key role that complex systems analysis should play in the design of future systems. Pervasive ICT will have vast numbers of individual components interacting together in extremely complicated ways. Complex systems analysis provides modelling and simulation tools that can provide a link between the design of the local rules and component interactions, and the resulting global behaviour of the whole system.

### Acknowledgements

The authors wish to thank other members of BT's Future Technologies Group at Adastral Park for their suggestions and feedback, which have contributed to the research described here in various ways.

### References

- 1 Waldrop M: 'Pervasive computing — an overview of the concept and exploration of the public policy implications', Publication 2003-3, Foresight and Governance Project, Woodrow Wilson International Center for Scholars — <http://wwics.si.edu/news/docs/pervcomp.pdf>
- 2 Horn P: 'Autonomic computing: IBM's perspective on the state of information technology', also known as IBM's Autonomic Computing Manifesto, IBM (October 2001) — [http://www.research.ibm.com/autonomic/manifesto/autonomic\\_computing.pdf](http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf)
- 3 Kephart J O and Chess D M: 'The vision of autonomic computing', IEEE Computer Magazine, pp 41—50 (January 2003).
- 4 'IBM and autonomic computing: an Architectural Blueprint for Autonomic Computing', IBM Publication (April 2003) — <http://www.ibm.com/autonomic/pdfs/ACwpFinal.pdf>
- 5 Chase N: 'An autonomic computing roadmap', IBM DeveloperWorks (February 2004) — <http://www-106.ibm.com/developerworks/library/ac-roadmap/>
- 6 IBM's Autonomic Computing Web site: — <http://www.research.ibm.com/autonomic/>
- 7 Tateson R: 'Self-organising pattern formation: fruit flies and cell phones', in Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Springer, Berlin, pp 732—741 (1998).
- 8 Tateson R, Shackleton M, Marrow P, Bonsma E, Proctor G, Winter C and Nwana H: 'Nature-inspired computation: towards novel and radical computing', BT Technol J, 18, No 1, pp 73—75 (2000).
- 9 Tateson R, Howard S and Bradbeer S: 'Nature-inspired self-organisation in wireless communications networks', in Marzo D, Karageorgos A, Rana O F and Zambonelli F (Eds): 'Engineering self-organising applications', Post-proceedings, Springer, Berlin (2004).
- 10 Erlanger L: '21st century security', Internet World Magazine — <http://www.internetworld.com/>
- 11 Saffre F: 'Adaptive security and robust networks', Information Security Bulletin, 7, No 11 (November 2002).
- 12 Anderson D P, Cobb J, Korpela E, Lebofsky M and Werthimer D: 'SETI@home: an experiment in public-resource computing', Communications of the ACM, 45, No 11, pp 56—61 (November 2002).

- 13 Bonsma E: 'Fully decentralised, scalable look-up in a network of peers using small world networks', Proc of the 6th World Multi Conf on Systemics, Cybernetics and Informatics (SCI2002), Orlando, No 6, pp 147—152 (July 2002).
- 14 Watts D J and Strogatz S H: 'Collective dynamics of 'small-world' networks', Nature, 393, pp 440—442 (June 1998) .
- 15 Kleinberg J M: 'Navigation in a small world', Nature, 305, p 845 (August 2000).
- 16 Kleinberg J M: 'The small-world phenomenon: an algorithmic perspective', Technical Report 99-1776, Department of Computer Science, Cornell University, Ithaca, NY (October 1999).
- 17 Marshall I W and Roadknight C: 'Provision of quality of service for active services', Computer Networks (April 2001).
- 18 Marshall I W and Roadknight C: 'Adaptive management of an active services network', BT Technol J, 18, No 4, pp 78—84 (October 2000).
- 19 Worden D: 'Understand autonomic maturity levels', IBM DeveloperWorks (February 2004) — <http://www-106.ibm.com/developerworks/library/ac-mature.html>
- 20 Shackleton M and Marrow P (Eds): 'Nature-inspired computing', Special Issue, BT Technol J, 18, No 4 (Oct 2000).
- 21 Kephart J O: 'A biologically inspired immune system for computers', in Brooks R A and Maes P (Eds): 'Artificial Life IV', Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems, MIT Press, Cambridge, Massachusetts, pp 130—139 (1994).
- 22 Gamma E, Helm R, Johnson R and Vlissides J: 'Design Patterns: elements of reusable object-oriented software', Addison-Wesley Pub Co (1995).
- 23 Bell J: 'Understand the autonomic manager concept', IBM DeveloperWorks (February 2004) — <http://www-106.ibm.com/developerworks/library/ac-amconcept/>
- 24 Hall T: 'Autonomic computing: it's about making smarter systems — an interview with Vaughn Rokosz', LDD Today, The Technology Journal for Lotus Software (June 2003) — [http://www-10.lotus.com/ldd/today.nsf/lookup/autonomic\\_computing](http://www-10.lotus.com/ldd/today.nsf/lookup/autonomic_computing)



Mark Shackleton graduated from Sheffield University in 1986 with a degree in Computer Science. He first worked for Singer Link-Miles, manufacturers of commercial flight simulators, developing real-time 3-D computer graphics algorithms and systems.

He joined the Image Processing and Computer Vision research group at BT in 1989. In this group he designed and implemented a number of systems in areas such as automatic face recognition, model-based coding, and content retrieval from images and video sequences. During this period he spent time seconded to MIT Media Lab working, closely alongside researchers there.

In 1996 he moved across to the Future Technologies Group, now part of the Pervasive ICT Research Centre, at Adastral Park. He now leads this group whose remit is to develop novel solutions for BT and its customers, using decentralised systems engineering approaches and radical technologies such as nature-inspired algorithms and complex systems modelling. The research is seeking to address issues of complexity inherent in the next generation of large-scale, complex, dynamic networks of computational devices.



Fabrice Saffre is a senior research scientist within the Pervasive ICT Research Centre of BT Exact and a member of the Institute of Physics. He holds a BA and MA in Philosophy of Science and post-graduate degree in Computer Science from the Université Libre de Bruxelles (Belgium), a post-graduate degree in Neuroscience from the Université de Nancy (France) and a science PhD (Theoretical Biology) from the Université Libre de Bruxelles. His primary areas of expertise are modelling and analysing complex systems, especially using numerical techniques (Monte Carlo

simulations and cellular automata), and the study of collective phenomena. His work in BT involves devising and evaluating nature-inspired solutions to a variety of problems encountered in distributed computing. These include routing, adaptive network security, robustness assessment and (the theory of) decentralised resource management (e.g. in P2P communities). He has authored many publications in biology of behaviour and complex adaptive systems (natural and artificial) in a variety of international journals and conferences. Since he joined BT, his work has generated a total of 9 patent applications.



Richard Tateson is a senior researcher in BT's ICT Research Centre at Adastral Park. He has a BA in Biochemistry from Cambridge University and a PhD in Developmental Biology, also from Cambridge University. Since 1997 he has been working on nature-inspired computation. His role is to identify and exploit natural solutions to telecommunications problems. Over the years this has involved drawing on cell biology, development, gene expression and evolution. His primary research interest is self-organising pattern formation in natural and artificial systems.



Erwin Bonsma graduated with distinction in June 1997 from the University of Twente with an MSc degree in Electrical Engineering.

He subsequently specialised in non-symbolic AI at the University of Edinburgh, where he obtained an MSc degree in Artificial Intelligence with distinction in September 1998.

In early 1999, he joined BT's Future Technologies Group at Adastral Park, which is now part of the Pervasive ICT Research Centre.

Here, he has been working on a multitude of projects, involving peer-to-peer networks, multi-agent systems, evolutionary computation and other nature-inspired techniques.



Christopher Roadknight graduated from the University of Manchester with an MSc in Computer Science and from Nottingham Trent University with a PhD entitled Transparent Neural Network Data Modelling.

He is currently a senior research scientist with BT's Pervasive ICT Research Centre at Adastral Park.

Having joined BT in 1997, he has worked on a range of artificial intelligence and future network projects.

These include the research and development of a WWW cache modelling tool-kit, a bacteria-based analogy for software placement on active networks and lightweight AI techniques for sensor network optimisation.