Making Autonomic Computing Systems Accountable: The Problem of Human-Computer Interaction

Making Autonomic Computing Systems Accountable: The Problem of Human-Computer Interaction

Stuart Anderson[1], Mark Hartswood[1], Rob Procter[1], Mark Rouncefield[2], Roger Slack[1], James Soutter[1] and Alex Voss[1]
[1]School of Informatics, University of Edinburgh
[2]Computing Department, Lancaster University
[1]*{mjh|rnp|rslack|jsoutter|av}@inf.ed.ac.uk*
[2]*m.rouncefield@lancs.ac.uk*

Abstract

The vision of autonomic computing raises fundamental questions about how we interact with computer systems. In this paper, we outline these questions and propose some strategies for addressing them. In particular, we examine the problem of how we may make autonomic computing systems accountable in interaction for their behaviour. We conclude that there is no technological solution to this problem. Rather, it calls for designers of autonomic computing systems to engage with users so as to undestand at first hand the challenges of being a user.

1. Introduction

The vision of autonomic computing, the automated management of computing resources [4], calls for the development of computing systems whose characteristics include:

- Capacity to configure and re-configure themselves under varying and unpredictable conditions;
- Constantly looking optimise their workings;
- Having knowledge of their environments and their context of use, and acting accordingly;
- Anticipating the optimised resources required to meet users' needs while keeping their complexity hidden.

This vision, we argue, raises fundamental questions about how we interact with computer systems. In this paper, we outline the nature of these questions and propose some strategies for addressing them. We wish to stress that we take a very broad view of what 'interacting with autonomic computing systems' means in practice. This view assumes, *inter alia*, an inclusive definition of the notion of user, interaction styles and timescales over which this interaction takes place.

We begin by introducing the notion of system accountability as a primary requirement for making sense of and trusting system behaviour. We then take an example of how accounts are typically provided in interaction and show why they are often inadequate. We consider some approaches to resolving the accountability problem in autonomic computing systems. We conclude by arguing for the importance of designers having a situated understanding of how current and future generations of computer systems are actually used.

2. The Problem of Human-Computer Interaction

Current paradigms of human-computer interaction exemplify the principle that users act and systems react. User actions are performed through the interface input mechanisms and their effects are signalled to the user as system state changes via the interface output mechanisms. Through this 'on demand' demonstration of a deterministic relationship between cause and effect, users learn to 'make sense' of a system's behaviour and so to 'trust' what it does. Increasingly, however, we find that this model of human-computer interaction fails to hold. For example, it is difficult for distributed systems, such as the WWW, to sustain observably deterministic behaviour in the face of e.g., unpredictable network delays.

Our earlier research shows clearly that users often demonstrate considerable resourcefulness in making sense of, and in coping with, this apparently non-deterministic behaviour [12]. This may often prove adequate for the purposes at hand, but the fact remains that many systems have ceased to be *accountable* in the ways in which current human-computer interaction paradigms presume. Being accountable means being able to provide an explanation for 'why things are this way' that is adequate for the purposes at hand. Accountability becomes even more relevant for systems whose role calls for users to make sense of their behaviour precisely and accurately [8]. Accordingly, researchers have begun to question of how systems might be made more accountable in interaction (e.g., [3]). The problem is that what 'counts' as an account is a situated matter.

We argue that accountability is a critical issue for autonomic systems. Further, we suggest that there may be different levels of autonomy and that there are attendant levels of accountability. In other words, there are

'grammars' of autonomy. In this paper we consider these two points as a preface to opening up the issues of 'sense making', 'trust' and 'repair' in autonomic computing systems. The issues we wish to address are these: what forms might autonomy take, how can it be made accountable and what character would that accountability take?

Drawing on our own studies of pre-autonomic – but still holdable-to-account – technologies [8,13], we will address issues of what types of information might be required to have trustable systems and how we might be able to integrate such systems into the fabric of daily life. We wish to examine questions around contingencies (how far must and how far can the world be 'tamed' to be suitable for autonomic computing systems?); the relationships of information, autonomy and context (does more information 'solve' the problem of context and how might this be employed in autonomic computing systems); and the ways that autonomic computing systems might be designed (can autonomy be a property of a generic system or is there a need to have bespoke systems?). In particular, we will examine the suggestion made by Dourish and Button [3] that we can treat accounts of system behaviour given by systems themselves as layered in character and thereby made relevant to different user populations and needs.

3. The Issue of Systems Management

We will take systems management as the focus for our examination of autonomic computing systems interaction issues. We may define systems management as the fine-tuning of system performance and configuration to meet specific organisational needs and practices, attempting to match these at all times to changes in the organisation's context. The point is that systems management as an activity may change, but will not disappear with the advent of autonomic computing systems.

Systems managers may need to be able to define and manipulate descriptions of organisational goals, configuration policies, to make sense of system performance data in terms of these descriptions so as to verify that goals are being met, and to identify, diagnose and correct goal failures. The interactional problems here are, in some senses, quite familiar. They stem from the difficulties of relating high-level, abstract system descriptions, as represented by organisational goals, configuration policies, etc., to low-level, behavioural data, with the added complexity of the likely dissociation of cause and effect due to inertia in system responses to configurational changes.

Our research into the static configuration of complex software systems suggests there are other issues with which we will have to grapple. For example, there is a tension between strict adherence to principles of 'good', standardised organisational practice and the needs that arise in local contexts of use [2,13]. Our research suggests that software systems architectures, and the additional overheads faced by systems managers, make this often very difficult to achieve in practice [2]. The intriguing question is whether autonomic computing systems offer a way out of this problem by eliminating much (if not all) of the management overheads associated with local configuration management, or make them worse because the system becomes more opaque and complex.

4. An Example of The Problem of Accounts

We now present an illustration of how, even with pre-autonomic technologies, it is difficult for systems to provide users with an adequate account of their behaviour. In Figures 1 and 2, we give an example of the use of layered accounts of system behaviour which demonstrates both how these have become commonplace in interaction and how they still fall short of

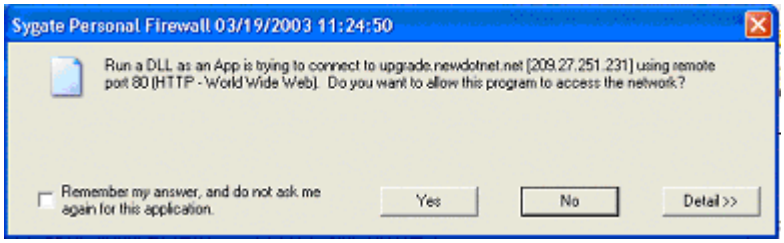

Figure 1: An example of a 'level one' account.

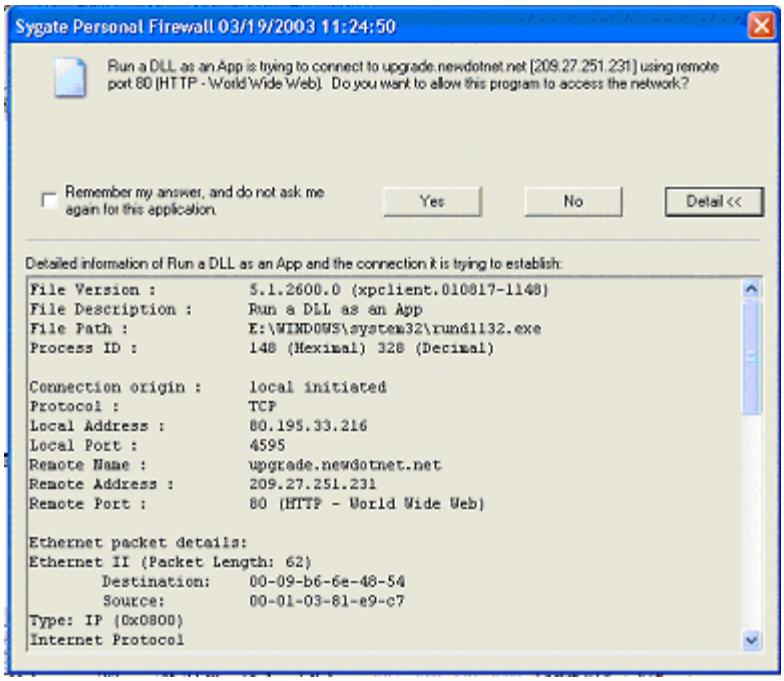

Figure 2: A 'level two' account derived from the level one account in Figure 1.

what the user actually needs in order to understand or to 'trust' the system. The example demonstrates the implementation of system security policy for potentially problematic Internet transactions. We develop this example to explore the implications of an autonomic approach to security policy implementation.

In Figure 1, we see what we call a 'level one' account of a situation in which the user is called upon to make a decision: a firewall has produced an alert concerning an application's attempt to access the Internet. Clearly, access to the Internet for downloading or uploading data to/from the system can be an accountable matter for system security. The firewall's implementation of policy with respect to these transactions is statically configured and involves calling upon the user to decide if access requests are appropriate and can be proceeded with.

Figure 2 shows the 'level two' account produced by the system in response to the user's request for more information. Note that the policy implementation requires user involvement since the firewall software cannot by itself determine if the transaction is to be viewed as a safe one. It is expected that the user has responsibility for deciding whether the transaction should be continued, and that the user has the requisite knowledge to decide whether the remote site is a trustworthy, whether the transaction is legitimate for this application and so on. Thus, implementation of the policy can be thought of as partial, requiring the user to 'fill in the gaps' on the occasion of its application. A key question is the degree to which it is feasible for automation to close these gaps. We return to this question later in the paper.

What the firewall system dialogues offer are accounts at varying 'levels' of description. The 'level one' account informs that there is some action required and gives some basic details. The 'level two' account furnishes further details about the nature of the transaction at a protocol level. The level two account is intended to help a user reach a decision about whether or not to allow the transaction to proceed, but unless the user has prior knowledge of the trustworthiness or otherwise of the given www site or is able to spot irregularities in the details of the transaction, then such accounts will be of little use. The account omits, for example, a description of *why* it was necessary for the application to make this particular transaction at this particular time. An improved basis for decision-making might be afforded by the firewall accounting for the context of the transaction: has the message appeared in response to the user's action or due to some background process? Is this a legitimate transaction for this application at this time? What are the potential consequences of the transaction?

One problem is that the 'designed for' layered account has a finite depth and extent; if, when the user has reached the last account, the explanation is still not adequate, the user is still unable to make a decision. Of course, it is relatively easy (technically) to supply accounts with increasing *depth*, but it is more difficult to increase the extent (or 'breadth') of the account, that is, to relate what the application might be trying to achieve, the implications of this in the context of the user's activity, location, and so on, until an account is produced that is relevant to the user's needs at that moment.

That there are many possible accounts that the firewall may provide raises the question of what sorts of accounts are likely to be useful. What our example accounts afford is "emer(gent) in the context of material encounters between actors and objects." ([9], p. 27). The question is, then, what do these accounts afford? An experienced system manager looking at these accounts might know what to do, but what of a more 'regular' end-user? How would such a user know what action to take, as well as what action the system was about to do? The firewall application provides a 'one size fits all' account, rather than accounts that are 'recipient designed' for particular users.

We should also note the role of organisational knowledge here: consider a user who knows what to do when she encounters a situation of this type. She will be able to take action based not on some concerted inquiry into the deep structure of the situation, but on some *sui generis* knowledge. Yet, there will also be occasions when this is not the case and when an understanding will be required. We will return to this point.

5. Autonomic Computing Systens and Accountability

5.1 The Stranger in the Loop?

"… automation reduces the chance for users to obtain hands-on experience; having been taken out of the loop, they are no longer vigilant or completely unaware of the current operating context. Thus, ironically, autonomic computing can decrease system transparency, increase system complexity and limit opportunities for human-computer interactions, all of which can make a computer system harder for people to use and make it more likely that they will make mistakes" ([10], p. 179).

This is at the heart of the problem of interacting with autonomic systems: autonomic computing systems are designed to work without users intervention most of the time, yet there is a sense in which users have to be in 'the loop' just enough to allow them not to be strangers to the system and what it is doing. We might think of this as the paradox of interactional satisficing, just enough involvement to know what is going on and what might need to be fixed without having to tend the system all the time. Yet, as in our example, it is only when things go beyond the rule set by which the system can make decisions itself that the system calls on the user. Should the user – in keeping with the aims of autonomic computing systems – have been getting on with other things, she is confronted with a system message into which she has to make a concerted inquiry.

5.2. Contextual Application of Policy and Accounts

One called for characteristic of autonomic computing systems is that they have access to contextual information, i.e., their operational environment and the activities of users. The question is, does such contextual information provide a solution to the problem of making autonomic computing systems accountable? There are two issues here. The first concerns whether contextualising autonomic computing systems makes their actions more reliable. The second is whether this can help shape accounts so that their actions are seen as more understandable, relevant and timely by their users.

To contextualise our firewall example in a simple way, a list of 'trustworthy' sites might be maintained for various sorts of Internet transactions. Rather than asking the user, access to non-trustworthy sites is denied. It may become more difficult to make sense of the system's behaviour, since some transactions of a particular sort may be allowed and others of the same sort denied. An account would need to be given in terms of the policy's implementation (i.e., listed and non-listed sites) for this to make sense. If the implementation is too restrictive, then it could frustrate users in their attempts to carry out their legitimate work. A less strict security policy might be to deny access to a known list of untrustworthy sites.

In order to mitigate some of these problems, one can envisage a more complex agent-based system that actively sought security information (from trusted sources) and maintained an access control list of known trusted and known untrusted sites. The system might also look intelligently for discrepancies in the transaction protocol for signs that a transaction may be untrustworthy. Although this may provide for a more specific policy implementation, it would also result in a system that which increasingly behaves in an apparently non-deterministic fashion (for some sites the system may deny access, for others it may allow access, and for a final group it may refer the decision to the user).

So, the availability of contextual information does not make the problem of providing accounts of system behaviour that the user can understand and trust go away; indeed, it gets more complicated. There are now three requirements these accounts must satisfy if the user is to be able to determine that the system's behaviour is consistent with that policy: they must provide a representation of the policy, some mechanism for providing evidence that will enable the user to trust that the policy is being conformed to, and some means of accounting for the system's behaviour in response to user demands.

Perhaps, however, if the contextualisation of autonomic computing system behaviour extends to knowing what the user is doing and what she intends by it, then this can provide a solution to an apparently escalating problem. In fact, there are two issues here. First, our studies show that organisational policies that are implemented without factoring in the user context are

likely to only make systems less usable and useful [2,13]. So, the decision-making context within which an autonomic computing system operates ought to be sensitive to what the user is doing, or intends to do. Role-based security policies, for example, are on attempt to incorporate the user context into system decisions. Here, organisational roles are used as rough and ready 'models' of users' access requirements. Arguably, such static user modelling techniques are better than none at all, but they are brittle in that they seldom capture the full extent of their application. The second issue concerns whether knowing what the user is doing and what she intends by it can be used to shape the accounts the system provides to its users. Here, the prospects are distinctly less promising. It is, as anyone who has had experience of so-called 'intelligent' user agents[1], is a very hard problem with, we argue, no foreseeable solution.

5.3 The limits of policy implementation

Policy cannot be enacted in each and every instance by the system, since there will be occasions when the system would not be able to specify what complying with the policy would be. That is, no rule specifies within itself all instances of its application [14]. There will, therefore, be times when the user is required to decide what action complies with policy. Just as a no entry sign means no entry on occasions when one is not driving a fire engine to a fire, but entry when one is, so a file might not be downloadable from a site on all occasions save for this one. That is, there are exceptions and humans are best at coping with these. Of course, there will be times when the exception does, in fact, become the rule and, again, it is up to humans to devise when this will be and to enable systems to make the required changes to realise this.

No system, therefore, can be wholly autonomic as there will be at some stage the need to have a human user input to decide policy and how to realise this – attendant to this is the need to keep the user informed about, *inter alia*, potential security threats, other changes in the environment and problems with fulfilling the security policy. This is not a trivial observation, since it turns on the accountability of systems, and the ways that they make problems, threats and shortcomings visible to users, and what users do about them. A problematic account might mean that a substantial amount of effort is required to 'excavate' the problem and formulate its solution. Therefore, when we talk about accountability we are talking about a pervasive and foundational phenomenon.

We are left with the problem of rendering the action (or inaction) of an autonomic computing system accountable as increasing complexity of the system makes its behaviour more opaque to end users. One solution might be to provide more complex accounts of the system's behaviour, why a transaction might be available one day, but not the next, on one machine but not on another, and so on. It behoves us to suggest what type of account we would add here, and in answer to that question we want to propose not simply one account, but a series of potential accounts linked to what, following [11], we call 'finite provinces of meaning'. That is to say, in recognising that there is no universal account that would 'do' for all practical purposes, we must develop a series of accounts that will do for the situated purposes – the finite provinces of meaning – that users might come up against[2].

Here, then, we turn to the notion of glossing. Glosses are "methods for producing observable-reportable understandings ... a multitude of ways for exhibiting-in-speaking and exhibiting-for-the-telling that and how speaking is understood." ([6], cited in [3], p. 16. Italics in original.) As Dourish and Button [3] point out: unlike machines, humans make their actions available in, and as a part of, their doing them – that is to say glosses are in vivo phenomenon for humans in a way that they are not for machines. We said above that accounts are constituent features of the circumstances they describe and are elaborated by those circumstances – this accountability is in part a gloss – after all one could not say everything about an activity to a person, there is always something more to say. Yet that does not mean that the gloss is opaque – no, the gloss is for all practical purposes here and now, elaborating and being elaborated by the things that it glosses. It is this that machines cannot do, as Dourish and Button [3] rightly point out.

Looking at the system accounts in the Figures 1 and 2, we find that they have been generated by a series of rules, rules decided by its designers when the system was created. These rules are to the effect that "if this or that happens put up this warning screen and make these choices available". Leaving aside for the moment the issue of how this could happen in a dynamic system, we might ask what use such an invariant account could be. This becomes especially important if one compares it with the activities of, say, a child on a merry-go-round: the child can tell about their experience as it is going on and do so in a number of ways that inform and are informed by the activity itself – that is, they can make the situation accountable in myriad ways. Therefore, one might ask the question "how do we get at these situated accounts?" Surely, a computing system cannot be expected to provide such accounts? We agree that it is problematic for a computing system – whatever its purported 'intelligence' – to produce such accounts.

[1] Microsoft's 'paper clip' is probably the most common example of this user interface technology.

[2] Indeed, we would argue that a substantial part of the problem in the examples given in Figures 1 and 2 is that they are designed to 'do' for all practical purposes.

Our solution is to be found in the examination of uses of a system and potential accounts that might come up, and to engage with users as to how the accounts might be designed so as to afford the kinds of information required. Users are not uniform, but when we look at policies and organisational arrangements we can see how 'what usually happens' and 'what is policy' might be resources to afford information, not to some idealised 'user', but to a 'member' – i.e., someone who knows what is going on around here and what are routine grounds for action and what are exceptions. We must, therefore, examine the constitution of membership by engaging with users in workplaces to develop accounts that afford what Gibson [7] termed 'information pickup' – i.e., information for the 'individual-in-the-social-context' as Anderson and Sharrock [1] put it. This also suggests that accounts should not be invariant – there will be a need to provide some information about the event within the account – but again, we argue that this can be realised through an engagement with users.

6. Conclusions

Autonomic computing systems will not eliminate the need for users to interact with them from a foundation of understanding and trust. In fact, as we have argued, they make this understanding and trust potentially more difficult to achieve. We might ask how far it would be intrusive (would we want to know what the system is doing at all times and, if not, when?). We do not need to know about the workings of the postal service to post a letter. Of course, we might want to inquire into these workings if a letter is late or undelivered. Yet, we would be unhappy if the post office called at midnight to say that our letter had been put into a train or the like. The point is that the system might be inquirable-into but it should not be obtrusive – that is to say, it should afford inquiry while not making operations obtrusive. In contrast, how far we would accept a more 'silent' approach (is the system working and what exactly is it doing?). This is an issue because it relates to trust; how can we trust a system when we are unaware of what it is doing? Trust is not an either or category, but depends on situated judgements which themselves turn on accountability, yet if the system does what it does in the background, so to speak, and, if it adapts, how can we know what dimensions are trustable?

As we have seen, solutions to this might involve the use of layered accounts that progressively divulge more information to the user on demand. They might also involve the exploitation by autonomic computing systems of contextual information as a means, for example, to guide its behaviour and for determining what account is likely to be called for by the user at any given moment. We have argued, however, that neither of these approaches can deliver a solution in themselves. All such 'designed for' accounts must have finite depth and therefore are limited in their capacity to answer what we take to be the user's canonical interaction question: 'why that now?' Similarly, contextualisation, in as far as it may be applied to the user's actions, can deliver little real benefit.

The overriding question is what do accounts generated by autonomic computing systems afford users, how might these accounts be assembled and for whom? We argue that it is through the consideration of these "seen but unnoticed" [5] issues that only comes from engagement with users that designers will be able to provide the kinds of accountability that users will need in order to make sense of, and to trust, autonomic computing systems.

7. References

[1] R.J. Anderson, and W.W. Sharrock. Can Organisations Afford Knowledge? Computer Supported Co-operative Work 1; 143-161, 1993.

[2] S. Anderson, G. Hardstone, R. Procter, and R. Williams. Down in the (Data)base(ment): Supporting Configuration in Organisational Information Systems. 1st DIRC Conference on Dependable Computing Systems, Royal Statistical Society, London, November, 2002.

[3] P. Dourish, and G. Button. On "Technomethodology": Foundational relationships between Ethnomethodology and System Design. Human-Computer Interaction 13(4); 395-432, 1998.

[4] A.G. Ganek, and T.A. Corbi. The dawning of the autonomic computing era. IBM Systems Journal. 43(1); 5-18, 2003.

[5] H. Garfinkel. Studies in Ethnomethodology. Englewood Cliffs, New Jersey: Prentice Hall, 1967.

[6] H. Garfinkel, and H. Sacks. On Formal Structures of Practical Actions. In J. C. McKinney and E. A. Tiryakian (Eds) Theoretical Sociology. New York: Appleton Century Crofts.

[7] J.J. Gibson. The senses considered as perceptual systems. Boston: Houghton Mifflin, 1966.

[8] M. Hartswood, R. Procter, M. Rouncefield, and R. Slack. Finding Order in the Machine: Computer-Aided Detection Tools in Mammography. In Proceedings of the 1st DIRC Conference on Dependable Computing Systems, Royal Statistical Society, London, November, 2002.

[9] I. Hutchby. Conversation and Technology. Cambridge: Polity, 2001.

[10] D.M. Russell, P.P. Magio, R. Dordick, and C. Neti. Dealing with Ghosts: Managing the user experience of autonomic computing. IBM Systems Journal. 43(1); 177-188, 2003.

[11] A. Schütz. The Phenomenology of the Social World. Evanston: Northwest University Press, 1967.

[12] D. Stanyer, and R. Procter. Improving Web Usability with the Link Lens. In Mendelzon, A. et al. (Eds.), Journal of Computer Networks and ISDN Systems, Vol. 31, Proceedings of the Eighth International WWW Conference, Toronto, May, 1999. Elsevier, pp. 455-66.

[13] A. Voss, R. Slack, R. Procter, R. Williams, M. Hartswood, and M. Rouncefield. Dependability as Ordinary Action. In S. Anderson, S. Bologna, and M. Felici (Eds.) Proceedings of the International Conference on Computer Safety, Reliability and Security (Safecomp), Catania, September, 2002. Lecture Notes in Computer Science v 2434. Springer-Verlag, pp. 32-43.

[14] L. Wittgenstein. Philosophical Investigations. Oxford: Blackwell, 1953.