<username>, I Need You!
Initiative and Interaction in Autonomic Systems

Piotr Kaminski
University of Victoria
pkaminsk@cs.uvic.ca

Priyanka Agrawal
University of Victoria
priya@uvic.ca

Holger Kienle
University of Victoria
kienle@cs.uvic.ca

Hausi Müller
University of Victoria
hausi@cs.uvic.ca

ABSTRACT
In this position paper, we examine factors, such as trust and usability, which can affect the adoption of an autonomic system. We argue that a system that exhibits initiative and strong communication skills is more likely to be adopted, and propose to treat humans as modeled, managed elements in an autonomic control loop to achieve these goals. We then propose some synergistic design ideas to make communicating with users more effective, and to allow the system to learn from the users' actions.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces.*

General Terms
Design, Human Factors.

Keywords
Autonomic computing, adoptability, trust, interaction, initiative.

1. INTRODUCTION
Autonomic systems are a technology whose time has come, their necessity precipitated by the ineluctably increasing complexity of our software constructs. There are still many engineering challenges to overcome: making a system self-configuring, self-optimizing, self-healing and self-protecting is no easy task. Yet perhaps the greatest challenge will come as the systems are deployed from research laboratories to customer sites. Will users gracefully accept granting computers a greater degree of autonomy? How can we ease adoption of this new paradigm?

First, autonomic enhancements must be truly useful to their users to have a chance of being used. Beyond this admittedly non-trivial requirement, however, we quickly run into the issue of trust—a relationship between two entities based on prior experiences and fulfilled expectations. Pre-autonomic systems had a relatively direct connection between the users' commands and the actions taken by the software. Building trust in the software was then simply a matter of demonstrating to the users that their commands would be followed faithfully and that any failures to do so would be considered as bugs and corrected. Barring excessively buggy software, users normally trust their computers to do exactly what they are told to do and little else. The situation can be likened to that of a master-slave relationship, where the slave's initiative is limited to the most menial tasks.

Engendering trust in an autonomic system is a much bigger problem [10]. The system no longer exhibits a direct cause-and-effect relationship between commands and actions. The connection is far more indirect, driven by declarative policies and knowledge bases, and often temporally disjoint. The situation is perhaps more comparable to that of a semi-autonomous subordinate, who takes standing orders but has a lot of leeway in their execution and the day-to-day management of its affairs.

When others raise the issue of trust in autonomic systems, they usually call for greater transparency and accountability [1], to allow users to audit the system's actions and figure out its "thought processes." We believe that these characteristics are necessary, but not sufficient to establish a solid trust relationship. In the next section, we identify sources of trust and present an alternative model for the user-system relationship that aims to ease adoption by making it easier to trust an autonomic system. We then extrapolate the model's design implications in Sections 3 through 5. Section 6 concludes with a summary.

2. INITIATIVE AND INTERACTION
Since autonomic systems are supposed to take over certain heretofore human duties, a thought experiment seems worthwhile: if you were to hire a person rather than an autonomic system to delegate the same repetitive yet critical chores to, what qualities would you look for [4]? Clearly, you would require competence in the skills relevant to the performance of the job, but you would probably also look for certain character traits:

- Unflagging concentration and attention to detail. Luckily, computerized autonomic systems already possess these attributes.
- Strong communication skills, to report status and issues accurately and concisely. You need to be kept informed since you are responsible for the subordinate's actions, without drowning in a sea of detail. You also need your subordinate to be attentive to your directions and to interpret them correctly, since a misunderstanding could result in the wrong actions being undertaken.

DEAS 2005, May 21, 2005, St. Louis, Missouri, USA.

- Strong initiative, tempered by an understanding of the position's boundaries. After all, there is no point in delegating a complex task if the subordinate asks for approval at every deviation from the norm. On the other hand, you do not want them exceeding the imposed boundaries for fear of having their actions cause (undesirable) side-effects that they are not equipped to predict.
- An understanding of their own limitations, and the willingness to seek help when those limitations are exceeded. Ultimately, this quality will determine whether the subordinate can be trusted, irrespective of their actual skill level.

We propose that embedding equivalents of these traits into an autonomic system will foster trust and enhance their acceptance by users. Our intention is not to make the autonomic system seem more human; rather, we are trying to abstract and transfer some desirable characteristics of human employees to their autonomic replacements, without seeking to hide the latter's intrinsic nature.

The traits proposed above, as well as their implementations discussed in the remainder of this paper, can be understood as concrete expressions of the accepted bases for forming trust between humans, namely rules, personal disposition, history, shared category membership, and roles [8]. For example, attention to detail and tempered initiative (never disregarding stated policies) demonstrate respect for rules. Strong communication skills include the need to understand personal disposition, so as to adapt interaction to the participants, and to keep in mind the interaction history (see Section 4). Finally, a system's understanding of its duties and limitations reflects the role it plays in the larger world.

Naturally, these ideas are not completely new. Much has been written about the importance of transparency in an autonomic system's decision process [3], and the need for an autonomic system to know itself [2] and remain faithful to the policies set by its users [5]. However, the emphasis seems to be on having the users understand how the system works, by exposing its deliberations and requesting approval for any non-trivial actions. At the same time, an insistence on making the systems "self-managed" relegates their users to a passive supervisory role. We do not believe that this approach is the most effective one for building trusted autonomic systems.

Instead, we think we should see users as partners, busy with their own problems but willing to lend the system a hand when approached appropriately. We feel that we should not shirk from letting the system take important decisions within its sphere of responsibility, for sometimes it is easier to ask for forgiveness than permission. Basically, we advocate the dual principles of initiative and interaction, for it is in the balance between these two that truly useful autonomic systems lie.

3. MANAGING HUMAN ELEMENTS

Traditional systems take a very passive view of their users. Often, the user is just an implied entity that hits the controls and (presumably) watches the screen. If necessary, the system might authenticate the user to load custom settings or derive appropriate authorizations, but this provides only a vacuous sense of identity. It is always the user that drives the interaction, and the system that responds. The closest most systems get to initiating interaction is raising a notification flag on the desktop or blindly sending a status email to a preset address.

To do better, we must first resolve the nebulous controlling entities into individual humans. The system must be aware of the people that interact with it by holding models of them that go beyond the usual username-password pair and personal preferences. Depending on the application domain, the system might need to be aware of the skills each person has and the roles they play in the relevant organization, as well as any relationships between the users (e.g., a corporate management hierarchy).

The model must also include a list of various communication channels that can be used to reach the person. Each channel should be annotated with data about its nature, including supported media types, communication delay and visibility. Some of this data can be built-in, some specified during system setup, and some gathered transparently during the system's normal operation as outlined in the next section. In any case, furnished with access to this kind of information, an autonomic system can intelligently get in touch with the appropriate person, tailoring the message to the recipient and the channel both to the message and recipient.

Matters can be improved further if the system has a behavioural model of how people interact. The model can initially be quite simple, containing knowledge that people do not like being bugged too often, may take a while to respond to a message (especially over weekends and holidays), and if they do not respond for a long time may not respond at all. Each model's parameters could be iteratively adjusted over time, as the system gains experience with interaction with each individual and with people overall. The behavioural knowledge would allow the system to make smarter interaction decisions, which might increase its trustworthiness.

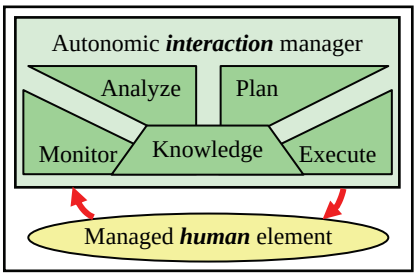

Figure 1. Managing a human

We now end up in an interesting situation, hinted at by Russell [9]. Our putative system can use its knowledge of people to plan its interactions with them, send messages to them, await their responses and analyze the results to refine its knowledge base. This structure is a perfect match for the canonical autonomic element architecture (as presented in [5] and adapted here in Figure 1) with the managed element replaced by a human being and our system playing the part of manager. By treating people as black-box, unreliable managed elements we can reuse the extensive research into autonomic managers, especially their modeling, learning and control capabilities. The four self-* properties of autonomic systems can also be reinterpreted in this context:

- self-configuration could mean automatically acquiring contact information (see next section);
- self-optimization could mean optimizing the flow of messages to balance disclosure against disruption;
- self-healing could mean recovering gracefully from changes in personnel and channel availability; and

- self-protection could mean guarding against malicious or incompetent users entering into the interaction loop.

In short, designing an autonomic system to treat people as modeled, managed elements has the potential to reuse existing autonomic structures and techniques to boost the system's interaction abilities, leading to increased trust and adoptability. With the overall framework established, the next two sections delve into the design of the communications themselves and the management of human responses.

4. MAKING EVERY MESSAGE COUNT

After deciding when and to whom to send a message, there still remain the important questions of how to send it, how to format it and what it should contain.

As mentioned in the previous section, if the system has a choice of channels for each person, it can select one best suited to a given situation. To help adoptability, it is important that the system support a wide array of communication channels, particularly ones that are already part of its users' daily information and conversation workflows [6]. Beyond the obvious (and much abused) technique of throwing up a dialog onto the user's screen, there are a few more suitable means, for example:

- Display an indicator on a local dashboard (such as the Windows system tray, or any of a number of fancier implementations). This channel is best suited for low-priority, continuously changing signals that the user can process in an ambient fashion.
- Post a message to an Atom feed (nee RSS) [7]. This protocol is best suited to broadcasting (or multicasting) messages that ought to be seen by many people. It is preferable to email since message streams can be aggregated and syndicated, and the messages are not vulnerable to email spam filters.
- Send an email message. Email is universally available and people have evolved elaborate idiosyncratic strategies for prioritizing and responding to messages. It is the primary means of communication on the Internet.
- Send an instant message (e.g., through MSN Messenger or Jabber). Instant messaging has become very popular over the last few years, and is a great way to contact someone when immediate action is required.

As an aside, note that many of the communication media above are at least somewhat asynchronous, in that the system cannot demand an immediate response from the user and may sometimes get no response at all. Accordingly, the system must be designed to continue functioning as best it can in the absence of an explicit external decision—a programming style that could prove difficult for developers used to blocking I/O and modal dialogs.

The presentation of the message should be adapted to its recipient, to increase trust based on the principles of personal disposition and roles. While the medium will shape the presentation to some extent, it should also be influenced by the system's knowledge of the user's skills, role and preferences. Naturally, this knowledge base will in turn be shaped by the user's responses (both explicit and implicit).

McLuhan notwithstanding,[1] the message content is important too: it should be adapted to the channel and appropriately succinct, while allowing the reader to drill down into details as needed. When possible, a message should offer the recipient the option of immediately selecting an action from suggestions relevant to the event being reported. To minimize the disturbance of responding to a message, the user's decisions should be conveyed in the same medium as the original message. For example, a blog entry (usually formatted in HTML) could embed a form to be posted back to the originating server, an email message could allow the user to take an action by replying with a coded subject, and an instant message could establish an interactive session as an impromptu command line. Staying within the medium eliminates a context switch for the user increasing the chances of getting a reply, and produces a strong association between the answering action and the causing event (which we take advantage of in the next section).

Nonetheless, a person may not feel compelled to respond to every message, so it is important that requests be easy to delegate. Whether by sending a URL, forwarding an email message, or inviting other people into an instant messaging discussion, a message must be shareable without losing its ability to select actions. Not only will such delegation increase response rates, for many media it will also implicitly bring new contacts into the system's social circle, for example by capturing a delegated responder's email address, or enticing them to subscribe to an Atom feed. Over time, the system can develop a better idea of the specialties of these contacts, and eventually get in touch with them directly when appropriate.

Even so, it is not our intention to elicit a response for every message. For notification messages, a lack of response can be interpreted as implicit approval, and for questions as implicit deferral to the system's expertise. On the other hand, if the system receives multiple conflicting responses to a message, it will need to further communicate with the concerned parties to clarify the issue. We do not expect the system to act as an arbiter in these circumstances.

In summary, improved methods of communication can lower the barrier of adoption and increase the response rate from users.

5. GROWING A POLICY

Asking a human contact for help could be seen as a failure of self-management, or a gap in the system's declared policies. Although it is preferable to ask for help in these cases rather than attempt to conceal the breakdown, it would be even better if the system could learn from the human's response to avoid such situations in the first place. We propose two closely related approaches, both enabled by the close connection between notification messages and the resulting actions taken by the human expert from within the message itself. Architectures without this connection—for example, traditional passive event logs with user action through a central interface—would be unsuitable for learning autonomic systems.

[1] Marshall McLuhan is famous for claiming that "the medium is the message."

The first approach is simply to store situation-action pairs in the knowledge base, and try to match an existing pair when a problem comes up. To be effective, situations (and hence messages) must be encoded in a declarative fashion so as to be easily searched for matches, precluding the direct generation of concatenated string messages in the application code. Furthermore, depending on the problem domain, it is unlikely that a new situation will precisely match a previously recorded one, necessitating the use of approximate matching. An optimization engine will need to balance the quality of the match, the desirability of asking for advice, the need to resolve the situation quickly, and the risk of taking the wrong decision. For example, if a medium-priority issue is encountered, there is a loose match against the knowledge base, the proposed action can be undone, and all relevant contacts have already been bothered recently, the system might decide to intervene autonomously and send a low-priority notification email. On the other hand, if the best choice of action is irreversible and the situation is urgent, or the user has recently countermanded another decision, the system might decide to seek advice from currently available relevant contacts via an instant message. The decision algorithm would need to be carefully tuned if the system is to garner the users' trust.

For domains where having the system guess the solution to a problem is not desirable (e.g., because all actions carry a high risk), another approach is possible. After a person takes an action in response to a message, they can explicitly expand the conditions under which the action applies, effectively creating a new policy rule for the knowledge base. This scope amplification can take place either in the initial event message, or in a follow-up message that is sent after the user has had a chance to evaluate the action's effects. In either case, amplifying actions should prove easier than manufacturing policy rules out of whole cloth, as the human mind is better at generalizing from concrete examples than at creating coherent abstractions from scratch, especially since the amplification options can be tailored to the specific situation and action selected.

One problem shared by both approaches is that the knowledge base will grow organically and could quickly become inconsistent and unmanageable. To mitigate this problem, each message that makes use of a policy rule—whether to notify of its automated use or to suggest its application—should also offer a selection of meta-actions to change, restrict, or rescind the rule, thus keeping the knowledge base in check. Nonetheless, the complexity of the knowledge base will tend to increase, so this and other problems will need to be investigated further before the proposed scheme can be successfully deployed.

6. CONCLUSION

In this paper, we argued that autonomic systems can become more trustable by actively communicating with their users. Improved interaction will also allow these systems to be more autonomous, exhibiting increased initiative without losing the users' trust. Higher trustability and usability should in turn lead to improved adoptability.

The design proposed to achieve these goals is to treat humans as external managed elements and leverage the machinery of the classic autonomic manager to control the system's interaction with its users. The system should take advantage of existing communication channels and include means for responding and delegating in every message. The system should learn from users' responses, possibly assisted by explicit action amplification, to reduce the need for user intervention in the future.

Although the ideas presented in this paper are still preliminary observations, we believe that they point the way to increasing user acceptance and adoption of autonomic systems. We intend to incorporate our ideas into prototype implementations of autonomic systems, followed by user studies to confirm whether our systems have a positive impact on adoption. We expect that the experiences gained by building prototypes and the feedback obtained by user studies will further refine our ideas, resulting in a comprehensive methodology for building adoption-friendly autonomic computing systems.

7. REFERENCES

[1] Anderson, S., Hartswood, M., Procter, R., Rouncefield, M., Slack, R., Soutter, J. and Voss, A. Making Autonomic Computing Systems Accountable: The Problem of Human-Computer Interaction, *Proceedings 14th IEEE International Workshop on Database and Expert Systems Applications (DEX 2003)*, pp. 718-724, Prague, Czech Republic, September 2003.

[2] Ganek, A. and Corbi, T. The Dawning of the Autonomic Computing Era, *IBM Systems Journal*, 42(1):5-18, 2003.

[3] Herger, L., Iwano, K. Pattnaik, P., Davis, A. and Ritsko, J. (eds). *Special Issue on Autonomic Computing*, *IBM Systems Journal*, 42(1):3-188, 2003.

[4] Kandogan, E., Maglio, P. P. Why Don't You Trust Me Anymore? Or the Role of Trust in Troubleshooting Activities of System Administrators. In *Proc. of the Conference on Human Factors in Computing Systems (CHI 2003)*, April 2003.

[5] Kephart, J.O., Chess, D.M. The Vision of Autonomic Computing, *IEEE Computer*, 36(1):41-50, Jan 2003.

[6] Müller, H., Weber, A., Wong, K. Leveraging Cognitive Support and Modern Platforms for Adoption-Centric Reverse Engineering. In *Proc. Of the 3rd International Workshop on Adoption-Centric Software Engineering (ACSE 2003)*, May 2003.

[7] Nottingham, M., Sayre, R., editors. *The Atom Syndication Format*. IETF Internet Draft, work in progress, January 2005. http://www.atompub.org/2005/01/10/draft-ietf-atompub-format-04.html

[8] Pyysiainen, J. Building trust in global inter-organizational software development projects: problems and practices. In *Proc. of the International Workshop on Global Software Development*, ICSE 2003, May 2003.

[9] Russell, D. M., Maglio, P. P., Dordick, R., Neti, C. Dealing with ghosts: Managing the user experience of autonomic computing. *IBM Systems Journal*, Vol. 42, No. 1, 2003.

[10] Telford, R., Horman, R., Lightstone, S., Markov, N., O'Connell, S., Lohman, G. Usability and design considerations for an autonomic relational database management system. *IBM Systems Journal*, Vol. 42, No. 4, 2003.