

Reaction Paper #4: “Usable Autonomic Computing Systems: the Administrator’s Perspective”

I immediately like how this paper presents the need and potential future use of ACSs in such a practical and applicable way: for the need of a system administrator. For me at least, it makes the issues and potential solutions seem all that more tangible. I also like how primary accounts from interviewed sys admins are interspersed throughout the paper, and that the accounts pertain to the particular section the authors (Bailey, Barret, Kandogan, and Maglio) seek to explain. Essentially, the authors conducted a seemingly large scale study among system administrators for large industrial computer service delivery centers in the U.S. and found that the task and goals of a typical sys admin can be broken down into 3 areas: 1) rehearsal and planning 2) situation awareness and 3) managing multitasking, interruptions, and diversions. Based on these 3 field-observed areas of system management, the authors argue that AC environments can be created to support and enhance some of these activities without “hindering the work of the administrators.” Only thing I didn’t like in this whole intro area was the fact that they claim early on in the paper, “AC *promises* to improve their [sys admins] work through automation.” Maybe I just don’t like empty promises. All in all, they had me interested since they were presenting such an applicable and practical viewpoint of AC.

The authors then go into the tools of network administrators, specifically, comparing and contrasting the current interfaces (CLI vs. GUI) admins currently have/use to diagnose and fix problems. They bring up pros and cons of both among the issues of: familiar/trained problem vs. unfamiliar/new problem, multitasking capabilities vs. single task at-a-time focus, peripheral information alongside info on primary task, reliable, detailed probing (e.g. lots of control to maneuver as seen fit in CLI) vs. more predefined set of diagnosable problems, support for graphics, taking the time to write custom scripts based on particular system to perform tasks, etc. They argue on one hand that the fact that some admins will take the time to create their own toolset for expediting the expansive set of components they must manage is, in itself, a sign that more (automated) help is needed. Then on the other hand, the authors follow quickly by pointing out that there already exist many automated tools that make “complex systems *less* reliable.” – therein lying the challenge of their proposed high-level-task-oriented AC for admins, ranging over the 3 areas listed above.

I like how the authors make case points for every section as to how that particular section’s focus on automation could bring further disastrous results than alleviate administrative problems; it shows that this paper is probably the most realistic one we’ve read thus far.

In section 3, “Rehearsing and Planning” is broken down into guidelines for which the AC managing tasks related rehearsal and planning should follow. I think for this section, the 2 ideas I like the most are the Undo feature as well as “graceful” automation changes. I feel as though the other aspects in this section the AC system should handle, such as *easily* building test systems that, to some degree of fidelity, simulate and run as the production system should, AND that it is consistent may be a ways off, and, in time could possibly be built on top of the manual “undo” and pre-approved automated changes...if the likelihood for consistent improvement outweighs the small chance for even bigger disaster.

In the “Situation Awareness” section, the very first thing I became confused about was in the statement that “[m]uch is known about providing appropriate awareness for automated control systems, but less about providing situation awareness for computing systems more generally”...? I am still not sure what the differences are, exactly. A good point is raised about the fact that for situation awareness being such a key attribute to successfully viewing/repairing

networks, as soon as automation steps in, you lose the vigilance and the incentive to maintain the alertness and overall view of the system(s) at hand...you begin to rely on an possibly unreliable machine to be aware for you. Our AI thus far is no where near sophisticated enough for a machine to replicate this cognitive ability in humans. Sure autonomic systems can and probably will be faster at remembering and diagnosing the bigger system as a whole, but when there are latent issues that build up within individual components, during a critical time the future awareness capabilities of the AC are not going to do much good. I agree that complex computer systems cannot be comprehended entirely as a whole to any 1 person, so perhaps the best that autonomic computing can do in the case of assisting and providing better situation awareness is to eliminate components of the system that are not at all related, thereby reducing the size of the system, and (as the paper also mentions) provide facilities allowing the administrator to GAIN further situation awareness of the problematic system (I don't know through what tools this would be done, but it sounds good at a high-level).

Multitasking, Interruptions, and Diversions – when I think of the problem here, without reading anything, I think of some sort of graph, which nodes as individual tasks, and the edges between them either as pre-requisites or paths on how to get to that task. Obviously with graphs, some flows are directional, some aren't, and some nodes have multiple paths and nodes leading to them. This to me DOES seem like something that autonomic computing can assist with, if by nothing else, graph theory and algorithms (Dijkstra's shortest path, critical path algorithms, etc.). This is especially true with a dynamic graph in which unsuspected problem nodes (interruptions and diversions) may be worked into the graph at any time. Then the autonomic system can somehow display the step(s) needed to proceed logically with the next task (node). Perhaps even advanced relationship models can/will be programmed in to sort out the relation each component has with one another – that way even if multiple consoles are needed to simultaneously multi-task a problem, the AC will control what comes up and prompts the admin on each screen, based on its relation code within the given context. Virtual remote desktops may end up being helpful in this situation as well – to switch between many different types of systems from 1 machine. Again, these are just my thoughts prior to reading anything in the paper on this section.

After reading the section on multitasking, interruptions, and diversions, this seems to be the section the authors are least sure about helping rather than hurting. It also seems to be the section most intimidating because of the daunting size of the type of autonomic system that could handle these tasks. I did not see any real sort of proposed, high-level ideas; mostly I saw concerns that implementing an AC to handle multitasking will in turn create more multitasking, and overwhelm administrators further.