# ;login:

THE MAGAZINE OF USENIX & SAGE

December 2000 • volume 25 • number 8


inside:


OVERVIEW:

NEEDLES IN THE CRAYSTACK:
WHEN MACHINES GET SICK



USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

# needles in the craystack: when machines get sick

## Part 1: In Sickness and in Health: The Three Laws

**by Mark Burgess**

Mark is an associate professor at Oslo College, and is the program chair for LISA 2001.

<*Mark.Burgess@iu.hioslo.no*>

In the early days of science fiction it was popular to write stories about how computers would go mad and destroy the world. To many computers users today, this prophesy apparently comes true on a daily basis, at least on a small scale. The fact that computers have taken over the world is, in a compelling sense, clear. Computers pervade our society from the kitchen to the supermarket. They are in our washing machines, our cars, our supermarket checkouts. They are responsible for monitoring and communicating the movements of a global economy, which, to a large extent, is stylized by the very computer technology it uses. They are in our televisions as well as our PCs, in video games and wristwatches. Soon we shall be wearing them and most likely even implanting them.

The fact that computers go mad and wreak destruction is also self-evident to anyone who works with them regularly. Computer programs are imperfect and they crash with astonishing regularity, destroying work and disrupting our lives. The word "mad," which is probably more sensational than necessary, is perhaps no more than a quaint metaphor, but the basic gist of this fictional vision has a wry truth to it.

The invention and proliferation of computers in our world has not merely simplified many difficult and mundane tasks. It has also changed the way society works. In spite of the now-comical prediction by IBM's Thomas Watson in 1945 that the world would only need five large computers, today it seems that they are as commonplace as plastic.

The fact that we rely on computers for so many of our daily activities has changed the way we live. Email is replacing pen and paper, secretaries are disappearing, we pay for items with plastic cards, we order goods from foreign countries just as easily as we do from the corner shop. We arrange our schedules according to which particular machine we have to wait for, and so on.

Because computer programs have limited capabilities, we often have to adapt ourselves to the software we depend upon. We change our work habits and reassess our wishes on the basis of what a computer can or is willing to do for us; we even work around faults (bugs) that we know to exist, bending over backwards for the machine. In a subtle but pervasive way, our use of computers controls us every bit as effectively as we are able to control them. In this sense, computers really have taken us over. It is not an evil computer intelligence that controls us, but rather something far more pernicious: we are self-styled slaves to our own almost religious use of The Machine.

Then there is the madness. Computers crash, they amplify small mistakes into huge mistakes, they do not always behave predictably, sometimes they work slowly for no apparent reason. As one popular email signature proclaims: computers allow us to make mistakes faster than any other invention in history. Sometimes they even seem stubborn in their refusal to do what we would like. They produce meaningless answers

In this series of essays, I want to take a broader, more reflective view of what it means to build and maintain our information systems.

to apparently sensible questions. Surely the behavior of something not entirely in possession of all of its marbles.

But enough of the melodrama. What is this all about? Why do computers not just continue to work like vacuum cleaners or ovens? They are, after all, just appliances, marginally more delicate, but machines nonetheless. Why indeed do cars break down, people get sick? Why do plagues upset nature, storms wreak havoc? Are there any lessons to be learned from these parallels? Why does anything stop working the way it is supposed to? Does it have to do with foreign invaders? Bacteria and viruses? Things jammed in the works? Bits that fell off? Sabotage by enemies of the state? If so, what are these disrupters and why do they exist? Is there a corresponding gremlin that invades computer systems, causing them to fail? One thing is certain: computer viruses are not the whole story.

In this series of essays, I want to take a broader, more reflective view of what it means to build and maintain our information systems. I mean to show that many of the problems we face in running the world's computer systems are not new, even though we seem to be learning the details now for the first time. As many of my physics colleagues often complain of their research: everything has been done before; every problem has occurred and has usually been solved in some other context. There are lessons to be learned from the past, from analogous situations and also from the writings of scientists and thinkers in many contexts. It is time to revisit these ideas and reflect on their virtues, as we wobble on the springboard of the new millennium, poised to launch forth through our hand-held mobile monoliths and be reborn as masters of it all . . . or not?

Why should anyone bother to make these connections? There are two reasons for this indulgence: one is that we might actually gain a greater understanding of the specific problems we grapple with on a daily basis and see solutions through a wider perspective. We are creatures of association, and the illusion of understanding is usually enhanced by our ability to appreciate the same idea in several contexts. The other reason is that reading about things we recognize in a new context can be inspirational and lead us in new directions. "My god, it's full of wildcards!"

Naturally, no one should be looking to copy blindly from analogs in nature or history. Cheats flunk their exams, and shallow comparisons lead us only into blind alleys. Visionary monoliths are not summoned forth by beating our computers over the chassis with Lucy's favorite thighbone (though the sentiment is sometimes compelling); and, whatever Clarke and Kubrick had in mind with their 2001 monolith, it was surely more than just a mobile phone, or a CPU with the pins broken off . . . but that does not mean that we cannot take a good idea and reinvent it.

Part of the reason that we do not immediately recognize old problems when they stare us in the face is that they are embedded in a novel context: wolves in sheep's clothing, or perhaps electric sheep on our Deckard's lawn, posing as red herrings. Displaced ideas must be adapted. The key to understanding and solving associative problems is abstraction: to distill the general principles from the specific instances and retain the lessons which they convey. These may then be molded into new forms.

There are two sides to be addressed to this interaction between humans and computers. We would control our computers, and yet our computers control us. We are locked in this loop. Our preferred approach to managing information systems is somewhat haphazard, though only a little more haphazard than our general approach to managing governmental and civic functions, or our daily lives for that matter. We strive for order,

through control, but this control is based on skeletal procedures, often without a sufficient regard for the whole. Usually we wait for failure, if we even plan for it at all, and then attempt to diagnose. At the same time, our practices are defined by the computers we would like to control.

Human dependency on computers is a weakness, which makes us vulnerable to their failures. When a key system fails, the machine stops, it no longer performs its function and the loop is shattered. Not only are humans dependent on computers, but computers are dependent on humans. Recently the dangers of dependency have been considered much more seriously in connection with the possibility of computer warfare. All one needs to do to paralyze a society is to cripple its computer systems. We have built a tower of cards, based on fragile machinery, ill-equipped to face even simple threats from the environment. This mutual dependency must be broken if one is to find a more secure relationship with machines.

Not only are humans dependent on computers, but computers are dependent on humans.

Interestingly, as is often the case, partial answers have already been explored by people who think up ideas for a living: science-fiction writers. During the 1940s, biochemist and science-fiction writer Isaac Asimov began to consider what kinds of problems would result from a society dependent on machines. Asimov's machines were robots, mobile computers with great physical strength, some of which had a human appearance. Asimov's future society was extremely reliant on robots. He quickly realized that machines would have the capacity to do great damage unless they were constrained with a tight leash. Asimov's answer to this problem was to endow automatons with a set of rules that curbed their behavior and prevented them from harming humans – in a sense, a theoretical immune system for humans against machines. Asimov and his editor, John Campbell Jr., together invented the Three Laws of Robotics. The laws went like this:

- A robot may not injure a human being or through inaction allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Asimov's laws were supposed to protect humans from complex machinery in unpredictable circumstances. It was clear to Asimov that people could quickly become dependent on machines, and thus there was a need to be able to trust them implicitly. Of course, he also knew that this can never be: no matter what kinds of rules one creates, there are always unforeseen circumstances which probe for loopholes in a set of rules, and many of his robot stories were based on the mysteries arising from these loopholes.

The main threat to any system working on a set of programmed instructions is that the complexity of its surroundings tests it in every conceivable way, picking at the locks until every door is opened. Sooner or later, the system will break under the pressure. As the character in *Jurassic Park* says, "Nature will find a way." My rule of thumb is: when the complexity of its environment exceeds the complexity of a system, the system will not be completely predictable. Nevertheless, a set of rules can, clearly, greatly reduce the risk of aberrant behavior in a mechanism.

Apart from protective protocols, like Asimov's laws, which protect quality of service, security against loss of service is easily secured by a strategy that is very familiar in Nature: *redundancy*. "Redundancy is the last avenue of our survival," wrote Robert

To understand why computers get sick, one needs to ask why they even work at all.

Silverberg of his post-biowar novel *Shadrach in The Furnace,* where the lead character is "system administrator" for Genghis Mao's biological system. In the biological world, simple mechanisms like single-celled creatures are not well suited to survival in complex environments, unless they exist in huge numbers. The only strategy a single-celled organism has against total and final death is to reproduce very quickly and maintain large numbers, before something steps on it. If one is destroyed, there will always be a backup to continue the execution of their simple program: *copy thyself.* For computer systems, one makes backups of data to prevent loss, uses multiple servers to secure failover capacity. The general rule is that we use *parallelism* rather than serial architectures to increase the assurance of delivery. This strategy is particularly important in complex environments, because environments (by their size and nature) exert a higher level of parallelism against the system than the system is capable of producing to protect itself.

Two key strategies, then, should drive computer management: mechanisms that protect against loss of service and mechanisms that protect quality of service. The extent to which such mechanisms exist varies. Certainly no off-the-shelf computer systems have such systems as a part of their basic design. Software and hardware design is obsessed with speed and convenience, not safety and reliability. The Volvo of computer systems has yet to be sold in showrooms, though it can be put together as a kit, with a little dedication. Since the mid-1990s several researchers, including myself, have argued for (and even built) experimental systems that take on different aspects of this problem.

Computers have come a long way since the punch-card looms from which they evolved. In a compelling sense, they are now very much like primitive organisms or societies. To understand why they are so unpredictable, one needs to piece together an understanding of how they work and of the nature of their environments. This is an involved story, but as a piece of commonplace technology, it is part of our cultural heritage and so it has earned the right to a few chapters of explanation. To understand why computers get sick, one needs to ask why they even work at all. Clearly something that never changes cannot get sick. So the first step is to answer the question: how do computers change? The journey there is not a difficult one but it is surely complex and fascinating.

The next part of this story will take us into the realm of technological development and our attitudes toward it.