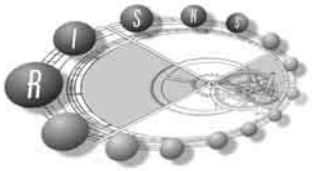

The Mindset of Dependability

COMPUTER SOFTWARE is legendary for its production time and cost overruns, and for its fragility after it is written. The U.S. government failed trying to procure dependable software for the IRS and the FAA, and the U.K. government was recently accused of wasting more than one billion pounds on failed or overdue information technology contracts. Perhaps only 25% of major software projects work out well. Home computer users are also accustomed to crashes. Why are computer systems so unreliable and difficult?

By contrast, the Japanese Shinkansen trains are a remarkable testimony to reliability and safety. Since their inception in 1964, carrying millions of people per year, no passenger has been killed as a result of a collision, derailment, or other railway accident. Not only are the Shinkansen safe, they are also reliable. The average Shinkansen train arrives within 24 seconds of schedule. What can be learned from this?

On one level, there are details of railway construction. The Shinkansen track is laid with heavier rail and closer-spaced cross-ties than a new line in Australia that will carry trains twice the weight.

On another level, safety benefits from Japanese culture. Any visitor can tell you that Japan is an extremely clean country; the Shinkansen tracks and stations are litter-free. The worst fire ever on the London Underground (King's Cross, 1987) started in debris under an escalator; cleanliness is not just cosmetic.

But historically, Japan was not renowned for railway safety. As recently as the early 1960s, just before the Shinkansen opened, two accidents near Tokyo each killed more than 100 people. And yet safety has now become routine. The culture of safety and dependability has been learned there; it could be learned elsewhere.

But *Communications* is neither a railway engineering journal nor a journal of cultural history. What should we learn about computers?

The Japanese did not do a cost-benefit analysis on safety. Nobody sat in the Shinkansen design office and thought about how to trade off cutting construction costs against the number of people that would be killed. In the computer context, we often distribute the costs of unreliable software over a great many users who do not easily aggregate their frustrations into economic impact. NIST recently estimated that software bugs cost the U.S. economy $60 billion per year. Lower testing costs, more features, and shorter time to market are easier to quantify than the benefits of various elements of dependability such as safety, security, and reliability—and may be viewed as more important by the development managers. If we care about having dependable systems, then we have to be sure that safety, security, and reliability are primary requirements. These are not things that can be patched in like an extra button in an interface. Today, vendors act as if people want more features and low prices first, and dependability later.

How can we achieve a culture of dependability? When buying a ticket to a symphony orchestra, people do not anticipate some particular percentage of wrong notes. Yet we routinely accept basic undependability in computer systems.

We have understood for a generation that having a small, terse, and limited system kernel greatly improves reliability. Yet we still see manufacturers resorting to special-purpose bypasses to make their particular program run faster or get around some blockage, with kernels swelling to tens of millions of lines of code. We still see complexity winning over simplicity.

How do we persuade manufacturers that security must be a priority? First, we have to believe it as users. People who routinely accept downloads from almost any site and use mailers that enable executable code attachments to send five-word ASCII strings wouldn't seem to care much about security or privacy. We need a culture change by purchasers as well as by developers. Perhaps the increased threat of cyberterrorism will reverse the trend of even security-conscious agencies to buy commercial off-the-shelf software without recognizing its risks; I hope it does so without any actual horror stories. Perhaps the recognition that simpler and more dependable systems can result in lower system administration costs, faster and fewer reboots, and lower training costs will help change the customer culture. If we can persuade manufacturers that more dependable software will pay off, and that adding more features won't enhance dependability, we might reverse a decades-long trend of greater vulnerability and lesser reliability. C

PAUL WATSON

MICHAEL LESK (lesk@acm.org) is the author of *Practical Digital Libraries* (Morgan Kaufmann, 1997) and currently works for the Internet Archive.