Software: Use at Your Own Risk

by Chuck Allison

"BANGKOK (Reuters)—Security guards smashed their way into an official limousine with sledgehammers on Monday to rescue Thailand's finance minister after his car's computer failed … All doors and windows had locked automatically when the computer crashed, and the air-conditioning stopped, officials said. 'We could hardly breathe for over 10 minutes … It took my guard a long time to realize that we really wanted the window smashed so that we could crawl out. It was a harrowing experience.'" [1]

> "Software development has been likened to nailing jelly to a tree—and for good reason."

Many of us merely reboot when software misbehaves, but, as one CNN report stated, "malfunctions caused by bizarre and frustrating glitches are becoming harder and harder to escape now that software controls everything from stoves to cell phones, trains, cars, and power plants." [2]

For years, users of software products have blithely glanced over license agreements such as the following: "[The Company] shall not be liable in any manner whatsoever for results obtained for using this software … THESE MATERIALS ARE PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY KIND … [THE COMPANY] AND ITS SUPPLIERS DISCLAIM ANY AND ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED."

Contrast this with the warranty for almost any other product you buy. When purchasing an appliance, the strength of the warranty is often what sways the buyer's decision. With software, it's always "use at your own risk." Is it any wonder we're conditioned to expect software to fail?

Is it really all that hard to produce software that works? Yes. Software development has been likened to nailing jelly to a tree—and for good reason. Is that an excuse for poor software quality? Not from where I sit. I think we are approaching a time when one-sided license agreements will no longer fly.

The same CNN article reports that "defects stem from several sources: software complexity, commercial pressure to bring products out quickly, the industry's lack of liability for defects, and poor work methods." The complexity stems from the increased applicability of automation to daily tasks and users' demand for the same. If there were more accountability for quality, however, vendors wouldn't rush to market so soon with a buggy product. Quoting the article again: "'Software is being treated in a way that no other consumer products are,' said Barbara Simons, former president of the Association of Computing Machinery. 'We all know that you can't produce 100 percent bug-free software. But to go to the other extreme, and say that software makers should have no liability whatsoever, strikes me as absurd.'" Users are getting mad as heck, and they're not going to take it anymore.

And so are developers and testers. How often have you been asked by management to cut corners? It is true that the natural tension between management and development is necessary to balance the forces that demand quality as well as profitability, but the nature of most organizations stacks the deck in favor of management in the wrong places. When that happens, quality takes a hit. A healthy organization will place management and development on a more equal footing as far as quality is concerned.

No document I'm aware of presents the case for moral principles in software development better than the code of ethics of the Association of Computer Machinery (ACM) [3]. Here is a sampling of its tenets:

- Contribute to society and human well-being.
- Avoid harm to others.
- Strive to achieve the highest quality, effectiveness, and dignity in both the process and products of professional work.
- Acquire and maintain professional competence.
- Accept and provide appropriate professional review.

Software may well deserve its bad rap because of a lack of integrity in the organizations and processes that produce it. Frederick Brooks' celebrated *Mythical Man Month* showcases the following quote from a fine French restaurant:

"Good cooking takes time. If you are made to wait, it is to serve you better, and to please you."

We need to be up front about the cost of quality software. Managers need to trust developers when it comes to technical matters, and customers need to trust that vendors are conducting business honestly. And that trust must be earned.

Someday, after the political, legal, and economic dust has settled around the business of software, we may enjoy what famed physicist Richard P. Feynman envisioned in an address to Caltech graduates more than thirty years ago: "the good luck to be somewhere where you are free to maintain … integrity … and where you do not feel forced by a need to maintain your position in the organization, or financial support, or so on, to lose your integrity."[4] **{end}**

References:

1] "Official Trapped in Car After Computer Failed." Reuters, May 12, 2003.

2] "Spread of Buggy Software Raises New Questions." CNN.com, AprIL 27, 2003.

3] www.acm.org/constitution/code.html

4] Quoted in *Feynman Lectures on Computation.* Perseus Publishing, 1996, p. 292.