A Matter of Trust

"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]

Many of us merely reboot when software misbehaves, but, as CNN recently reported, "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 we users of software products have blithely glanced over license agreements such as the following, taken from an actual product on my shelf: "[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 just about any other product you buy at the local retailer. When purchasing an appliance, the strength of the warranty is often what sways the buyer's decision. With software, it's always a "use at your own risk" scenario. 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 the poor state of 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 CNN one last time: "Others say bugs would be greatly reduced if software makers were held legally responsible for defects. '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.

So are the many developers who are true to their Inner Programmer. How often have you been asked by management to cut corners, to not do the quality job you know needs to be done? The ethical programmer will endeavor to make a case for quality anyway. No document I'm aware of presents the case for morals in software development better than ACM Code of Ethics. It contains pertinent principles to live and lead by. I would suggest that all developers and managers become familiar with it[3]. Someday, after the political, legal, and economic dust has settled in this matter, we may perhaps enjoy what famed physicist Richard P. Feynman envisioned in an address to Cal-Tech graduates almost 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]

---

[1] Reuters news article released May 12, 2003.
[2] "Spread of buggy software raises new questions", CNN.com, Apr. 27, 2003.
[3] See http://www.acm.org/constitution/code.html.
[4] Quoted in *Feyman Lectures on Computation*, Perseus Publishing, 1996, p. 292.