Instability and Faith in the Art of Computer Programming

I confess—I have a preference for a certain electronic tablet and smartphone. I am not going to give its name, but suffice it to say that over the years I have learned to rely on it for more uses that I thought possible. It comes with 3G connectivity so it allows me instant access to the Internet almost anywhere. Once you start using one, well, it becomes part of your life.

Need an address? Contact list. Need a location? Maps. Want to see who was the stunt double for Judy Garland in "The Wizard of Oz?" Use IMDB. (It was Bobbie Koshay. And, for you true Oz fans—yes, I know—Caren Marsh-Doll was the off-camera stand-in, but NOT her stunt double.)

The problem is, you begin to depend upon these mental crutches. I no longer remember phone numbers—why bother? Let me use these wonderful appliances to push back the effects of senility and dementia just a little bit longer. I rely on these mental crutches so much so that my daughter refers to it as my, "second brain."

Until, of course, they do not work. I am not talking about the typical, "I am out of range, let me hope I find a signal soon." I am talking about the feeling I recently had when an update to the software on both my phone and tablet caused my formerly rock-solid browser to crash. Frequently. Every few minutes. All the time. Constantly. All the time.

While at my parents' house recently, the only Internet access available was my 3G tablet/phone. However, I was unable to reserve a hotel room for the return drive. On an earlier trip—I could not book an airline ticket. I was not even able to enjoy my morning ritual—coffee and browsing the Internet.

Now, I am not saying that my life was a catastrophe. I am just saying that instead of depending upon connectivity and modern technology, I (GASP!) had to make phone calls. Talk to real people. Read and touch an actual newspaper. And—when a recent update to the software fixed the problems—I somehow still seem to still have a bit of mistrust in my connectivity issues. It is just a little bit harder to take it for granted. I have been burned, and sometimes burns take a while to heal.

Which got me thinking. We have had a few depressions/recessions in the past 100 years. And if you talk to an economist, they will tell you that one underlying cause is consumers who have lost faith in the stock market.

Is this where we are heading in the computing age? I talk to more and more users who complain about lack of faith when using applications.

There are some software applications where we absolutely cannot have lack of faith. Pacemakers. Reactor software. Braking software in cars. Aircraft control software. And the list goes on and on. As more and more large-scale applications are used worldwide, the need for more and more faith in the software becomes a necessity.

I currently teach Software Engineering, along with Requirements Engineering and Systems Design. And, as part of both courses, I cover some really spectacular software failures (see <http://spectrum.ieee.org/computing/software/why-software-fails> for a great list). There is no shortage of cases to discuss. Some with loses of billions of dollars. Some that caused lost of life. Some that caused large companies to simply go out of business. And in almost all of the cases, one of the common factors tends to be lack of user involvement. Some might find it simply amazing that when writing software that might cost millions and millions of dollars there is a lack of user involvement. However, to experienced practitioners of the craft of producing software, it is not amazing at all. Bizarrely, the more software seems to cost, the less concerned users are with involvement in its creation. It is like the mere act of spending lots of money somehow relives them of the responsibility of being…well….responsible. In fact, for really large-scale software, it is often next to impossible to actually identify who the users are!

I have been teaching software engineering since the 1970s. Back in the day, we had the waterfall model. We realized, of course, that nobody really followed it—but it was what we had. That is, until the Spiral Model came along. And we realized that everybody really did it over and over and over again until they got it right (or, if not right, at least usable).

But the large-scale models did not adequately address the constant and continuing needs for more user involvement, getting the requirements right, getting a design that will be usable, uncovering the missing functionality, making sure that users' needs and wants are met, and ensuring the validity of the software before it is released to the users.

Is agile and/or rapid development the cure to all of our ills? Does it guarantee the correct software? Does it ensure validity? Does it make the users happy?

No, of course not. It is just a technique. But, it is a technique that focuses on smaller increments. It also focuses on having user involvement frequently, and in time to gently guide the development process so that it has a better chance of delivering software that the user will have a stake in.

And it will help in delivering software that users might just have a little more faith in.

David A. Cook, Ph.D.
Stephen F. Austin State University
cookda@sfasu.edu