Software Testing: What Goes Around Comes Around

Jeffrey Voas, Associate Editor in Chief
Keith W. Miller, University of Illinois at Springfield

Over a recent dinner, we compared notes and revisited a subject near and dear to our hearts: software testing. We shared similar concerns about how little today’s technology gurus understand about this most fundamental software engineering pillar and the challenges it presents.

Software Testing
Most people knowledgeable about technology generally understand that testing is useful, but they’re short on concrete details as to why. Furthermore, some fail to recognize that testing can be as problematic as it is beneficial. Some think that software testing is merely a set of tools—they have no idea about its rich history of debates over deep theoretical questions. Furthermore, they’ve never pondered its important relationship to mathematics.

Many people with an otherwise sophisticated understanding of technology don’t fully grasp the basic concept of software testing—that it’s traditionally performed to either assess a characteristic, such as reliability or security, or estimate the density or number of code defects. Furthermore, many don’t know the benefits or weaknesses of static versus dynamic testing, nor when to apply one over the other. Even more astonishing is that these are the same folks who ask, “Why isn’t software certified?” They ask this as if certifying software were as simple as pushing an elevator button.

Following are some of the key software testing questions we must consider:

- How do we know when to stop testing?
- What type of testing should we perform—manual code inspections or unit, system-level, black-box, white-box, or integration testing—and in what combinations?
- What should we test—reliability, performance, security, or something else?
- What tools should we use—commercial or open source?
- What defects should we fix and why?
- When should we retest?

Until we can satisfactorily answer these questions, there will be people who view software testing as more a craft than a science.

Mobile Apps
So why did these long-standing (and geekish) problems become our dinner topic? We started on this conversation after discussing the “buzz” about mobile apps and the role of testing in this new worlds of apps. This led us to another set of questions:

- What types of testing would we use for apps, and how much testing is enough?
- Is it easier to vet the app vendor or test the app?
- What will we be able to say about an app’s quality from inclusion in a particular app store?
- If app stores become software quality gatekeepers, will they be able to filter out lesser-quality apps quickly and in near real time?
- If app stores become the de facto filters for app quality and security, who will certify that the app stores are doing their task adequately—for example, disallowing counterfeit apps?

In short, trust shouldn’t mean something different for apps than for any other type of software.

Until we gain a better understanding of these questions and their relevance to the volumes of consumer app software being written today, we’ll inevitably be relying on poorly tested code—which can expose intimate details about our lives as we continuously use the apps. Patrick May of...
San Jose Mercury News sums it up as follows:

Federal and state regulators, along with privacy advocates, are pushing for more clarity and transparency in the way apps may use personal information, including your name, gender, and email address, as well as your hometown, family relationships, or religious and political affiliations. Various versions of a so-called ‘privacy bill of rights’ for mobile phone users are circulating and being adopted by some app developers.

So how can the average person, unfamiliar with apps, understand how much privacy is being given away or contemplate even less obvious concerns such as the amount of power being consumed? Not everyone knows that apps can drain batteries quickly.

It seems odd, if not reckless, to open ourselves up to the security and reliability risks that mobile apps might incur. So why are we doing it? One reason is that it’s convenient to ignore the risks. Furthermore, the alternative would be to abandon apps—something few of us are willing to do.

Is there hope of changing this situation soon? It depends. In industries where public safety is an explicit, central concern, there has been progress in developing, following, and in some cases mandating processes designed to enhance software safety.

However, the kinds of measures taken to improve software quality in, for example, nuclear power plants or airplanes, aren’t inexpensive or fast. Software developers in these industries might grumble, but they’ve come to expect to pay substantially to test up to the standards mandated by, for example, the US Federal Aviation Administration. The culture in these industries has changed as well, through regulation and a sincere desire to avoid harming the public with software.

Each piece of software used in aircraft, medical devices, or nuclear power plants is part of a complex set of relationships and artifacts—a sociotechnical system (http://stsroundtable.com/wiki/Socio-technical_systems). These systems have been tuned to enhance safety and reliability. Alas, the sociotechnical systems around mobile apps have been tuned to enhance affordability and convenience, and security and reliability appear to have been deemphasized to the point of near invisibility.

This leads us finally to the issue of certification. Will there ever be the equivalent of an Angie’s List (www.angieslist.com) or “Computer Reports” (www.consumerreports.org) or Underwriter’s Laboratory (www.ul.com) for apps to answer at least some of these questions about their quality? If so, why would this appear now? After all, we didn’t have it for the software that predated apps. It would be optimistic for the world of apps to soon develop a culture of lower-risk, higher-cost software based on sophisticated testing methods and certification.

Given a choice between high-cost certified apps and low-cost uncertified apps, most consumers would probably select lower costs. But at the moment, we don’t have that option.

Acknowledgments
This editorial reflects Jeffrey Voas’s personal opinions.

References

Jeffrey Voas is an associate editor-in-chief of IT Professional. Contact him at j.voas@ieee.org.

Keith W. Miller is the Scheewe Professor of Computer Science at the University of Illinois at Springfield. His research interests include computer ethics, software testing, and online learning. Contact him at miller.keith@uis.edu.

Selected CS articles and columns are available for free at http://ComputingNow.computer.org.