Software Testing and Industry Needs

An IEEE Software *special issue is usually rounded out by a point-counterpoint discussion in which two experts take opposing views. We've opted for a variation of this approach by inviting five renowned experts in software testing to give a brief answer to an admittedly broad question:*

Does the practice of software testing effectively meet industry needs?

—Natalia Juristo, Ana Moreno, and Wolfgang Strigel

Proof of the Pudding

Robert L. Glass

Is software testing practice meeting industry needs? I wish I knew!

Forgive my facetious response, but understanding the state of any software practice is difficult. I guest-edited a special issue of *IEEE Software* on the general topic of software engineering practice in 2003. I received a good collection of useful research findings, but I can't forget one response from a prominent computing practitioner-researcher: "There is no such thing as the state of the practice." What she meant was that practice is so varied that it's difficult to generalize.

That's certainly the case with the state of software testing practice. As one of the articles in this issue states, there might be a common general understanding of what testing is, but practices vary tremendously across companies.

But that's a cop-out. On balance, I actually do believe that most software testing in industry effectively meets industry needs. I base my bias on the observation that this is the computing era—an era that wouldn't be successful without successful software—and on my belief that software can't be successful without effective testing.

Does that mean I believe that testing practices can't be improved? Of course not. One article in this issue notes that practitioners aren't using test coverage measures very much, and that's a big problem. There are many ways of doing inadequate testing, and my guess is that—as we speak—all of them are being employed!

Still, the proof of the pudding is in the eating. Most software projects, I would strongly assert, produce products that do what they're supposed to do. And that wouldn't happen if testing were ineffective.

Robert L. Glass is a visiting professor at Griffith University, Brisbane, Australia, where he works in the Australian Research Council Center for Complex Systems. Contact him at rlglass@acm.org.

0740-7459/06/$20.00 © 2006 IEEE

A Career Quest

Ross Collard

Are software testing practices effective? To use the tester's infamous answer: "It depends."

My guess is that 65 percent of knowledgeable practitioners assess current practices as mediocre, with 25 percent more assessing it as negative, leaving a fringe of optimists. The reasons are many, have been listed before, and encourage debate about "how good is good enough?"

Let's look at hiring and career development as an assumed bellwether for all testing practices. Usually omitted from test practices lists, this area might not measure the overall state of testing, but it surely constrains its effectiveness.

To overgeneralize, the practice in these areas is often mediocre. A recent job ad listed these required skills: LoadRunner, Java, XML, WebSphere, Oracle, UML. Their relationship to testing is unclear. I suppose that skepticism, perseverance, clear thinking, and attention to detail are assumed or too amorphous for a job listing.

External perceptions of testing's value are mixed. Most people still enter the field accidentally. Test jobs are often consolation prizes for those not considered good enough to hire as software engineers, and the salary gap between developers and testers remains significant. Testers frequently complain about lack of respect, credibility, and influence. Career development is more haphazard than strategic.

Most testers are self-taught, and many have never read a book on the subject. No universally agreed-on terminology and testing body of knowledge exist. University training is widely second-rate. Major test certification programs could endanger testing's future by focusing on the wrong things.

So would you recommend a testing career to young people? I would. Why?

Many testers love the work and wouldn't trade their careers for anything. Despite (or perhaps because of) the complications and misperceptions, testers have unusually strong opportunities to contribute to a society that increasingly depends on reliable software. Effective testing is a quest and, like any quest, includes intellectual challenge, passionate debate, and the excitement of discovery. Rewards for the best 20 percent of testers typically include exponential career growth, substantial monetary compensation, and wide influence. There is plenty of room for more good testers.

Ross Collard teaches, writes about, and consults on testing. Contact him at ross@rosscollard.com.

Closing a Gap

Antonia Bertolino

While everyone will concur that a large gap exists between software testing research and industry practice, there isn't agreement regarding reasons, responsibilities, and possible remedies.

Let's take coverage testing as an example (that is, targeting test cases to thoroughly exercise a program's structure). Researchers actively investigated coverage testing until the mid-1990s. To what effect? Ask researchers about it, and they will be familiar with a spectrum of white-box testing criteria based on control flow or data flow and elegantly ordered into a subsumption hierarchy. Ask practitioners, and in the best case, they'll know about branch coverage; more commonly, they'll take coverage testing to mean "statement coverage."

The essence of software testing is in systematically sampling behavior, and the effort of research is in finding effective means to pursue this systematicity—the many coverage criteria rely on code exploration to provide different thoroughness levels. But proposed research solutions still call for substantial investment before they can be put to work. Proof-of-concept prototypes can suffice to demonstrate the idea in a paper, but coverage testing, for example, still needs sophisticated tools for code instrumentation and monitoring as well as empirical assessment of relative metrics.

This is true for any research field. What is it that makes deployment of research results particularly difficult in testing? On one side, practitioners who are chronically short of time or resources tend to perceive systematic testing as a luxury. On the other, testing impacts the whole life cycle, because any testing technique presupposes adequate preparation, modeling, and documentation.

So, there are two challenges for filling the gap, and only researchers and practitioners working closely together can meet them: promoting a mind-set change regarding the intrinsic value of systematic testing—that it's not a cost but an advantage—and recruiting adequate investments for transferring research results into practice by integrating systematic testing techniques seamlessly into the development process.

Antonia Bertolino is a research director at the Italian National Research Council and leader of the Software Engineering Research Laboratory at Istituto di Scienza e Tecnologie della Informazione "A. Faedo." Contact her at antonia.bertolino@isti.cnr.it.

Beware the Pundits

James Bach

There is no one practice of software testing. Instead there are many practices and practice communities. Nor is there any one context against which to judge testing's effectiveness. The behavior that works well testing video games at Electronic Arts would fall apart when testing flight-planning software at Eglin Air Force Base. Still, I see an underlying skill that all people who strive to be great testers should develop.

This skill is better known as general systems analysis or general systems thinking. By and large, this skill isn't taught, nor is it even acknowledged by most people who give opinions about test processes. In that respect, I think industry is not being served by most of the pundits who consult and write about testing practices.

If more testers actually read testing textbooks or paid attention to standards, this would be a big problem. However, most testers ignore most of the advice that's being offered about testing practices, so the fact that it's mostly bad advice isn't doing much harm. I think IEEE 829 is a bad standard, for instance, so I'm happy that the IEEE doesn't post it freely on its Web site for anyone to find and use. Few testers in industry even know about it. In this case, I think ignorance really is bliss.

Industry ultimately takes care of its own practices. Each company does what it believes will work. What we need to resist—and resist strongly—are efforts to take away each company's right and responsibility to set practices for itself. Good practice cannot be legislated from a distance, any more than a doctor can diagnose a patient sight unseen. Industry-wide licensing and certification efforts haven't and won't advance good testing practices until and unless the pundits who push them, in the same way doctors who treat patients, are made fully accountable for the advice they give.

James Bach is an independent consulting software tester. He teaches rapid software testing skills through Satisfice. Contact him at james@satisfice.com.

Investigation, Rather than Control

Cem Kaner

The practice of software testing is evolving oddly. University research fails to inspire many practitioners. Advances in industry practice have little impact on research. In the meantime, programmers' productivity accelerates relentlessly faster than testers', threatening to render testing irrelevant because the productivity gap means testers impact proportionally less code each year.

One effect of this productivity gap has been a rekindling of interest in unit testing. The evolution of agile programming, such as Extreme Programming, is partially the story of programmers who despaired of getting much more than endless paperwork from testers and chose to assert primary responsibility for the implementation quality of their own code.

Another trend among practitioners has been evolution away from quality control toward quality assistance. Under this alternative view, which I share, software testing is an empirical investigation, conducted to provide stakeholders with quality information about the product or service under test. Note the emphasis on purpose instead of practices. Most industrial projects have many stakeholders who have diverse interests and conflicting priorities. The purpose of testing during development is to help those stakeholders understand what they're getting, in time to correct a weak programming practice or (re)negotiate the design.

The technical challenges of testing grow not just because programs are bigger but also because we expect software to run correctly even when the code is multithreaded, when the data are passed back and forth in real time across distributed systems, and when the software runs for longer and longer times without reinitialization.

Under these conditions, testers are measured by their skill as investigators and communicators and by the tools they can create and use to support their investigations—not by their level of control over the product's code or design.

Not much guidance for these efforts exists in current software engineering standards, academic research on testing, or the tester certification businesses that have become so popular in this decade. For software testing to meet industry needs, this will have to change.

Cem Kaner is a professor of software engineering at the Florida Institute of Technology. Contact him at kaner@kaner.com.