DevelopSense Newsletter **March 11, 2008**

61Ashburnham Road Toronto, ON M6H 2K4 (416) 656-5160 info@developsense.com

So maybe you're wondering where I've been. Some people who signed up for my newsletter haven't heard from me for a year or more. Some people haven't *ever* heard from me. Well, it's pretty simple: I've been busy, teaching, writing, learning, reading, presenting at conferences, setting up conferences, and attending conferences. The sun never sets on Rapid Software Testing. There are lots of good things about that, and some bad things too—absence from family is particularly tough. But I've met a lot of testers and enjoyed a huge number of fascinating conversations. (I've also played at Irish traditional music sessions in pubs all over North America.) All this has amounted to me not writing or sending newsletters. The blog (http://www.developsense.com/blog) is one way of keeping in touch. But until now, I haven't felt I've had something sufficiently newsworthy to send out a newsletter.

Now I do, though. Here it is:

CAST Comes To Toronto: We want YOU; we need SPONSORS

The Third Annual Conference for the Association for Software Testing is happening in Toronto, July 14-16. It will be *amazing*.

Let's start with the normal, routine, annual stuff. CAST is a special kind of conference, because it's about *conferring*. The presentations work differently from everywhere else. After each presentation, there is a discussion immediately following, guided by a facilitator who helps to make sure that everyone can be heard and noisy folks (like, uh, me) are contained. The conversation continues as long as there's energy for it—and if something is generating a lot of heat (and ideally light, though that sometimes comes later), we shift the schedule or make some other change to accommodate the flow of conversation and ideas. No other testing conference of its size does this (and believe me, I've been to almost all of them in North America, and a good number of them in Europe and Australasia).

Second: should some calamity happen at the conference, most of the public face of the testing business—especially the context-driven crowd—would disappear. The most astute thinkers, the most talented trainers, the most articulate bloggers, eloquent writers, closest colleagues and contentious combatants—often all at the same time, in the same body—are regular attendees, and the many of the names missing from the attendance sheets in past years will appear this year. And that's just the noisy ones. More and more members of the worldwide testing community are participating.

That's the conference and the community. But here's the kicker: the theme of the conference is "Beyond the Boundaries: Interdisciplinary Approaches to Software Testing" and keynote presentations will be given by **Jerry Weinberg, Cem Kaner, and Rob Sabourin.**

Jerry's keynote topic will be *Lessons from the Past to Carry into the Future*—and what a past Jerry has had! He's been in the business of software testing for more than 50 years—he was one of the programmers on Project Mercury, which sent Americans into space for the first time. Jerry is the author of more than 30 books, several of which are cornerstones of my list of books that testers should read. He's also the co-creator of the legendary Problem Solving Leadership workshops and the Amplifying Your Effectiveness conference.

Cem Kaner should be no stranger to the readers of this letter, either. He's the senior author of two of the best-selling testing books in history, the senior author of the free Black Box Software Testing course, and one of the all-time great teachers of software testing. He's the co-founder of the Los Altos Workshops on Software Testing, and thereby the godfather of all of the LAWST-style peer conferences, including the Workshops on Teaching Software Testing, the Software Test Managers' Roundtables, the Workshops on Heuristic and Exploratory Techniques. He has a doctorate in psychology and a degree in law. In his talk, *The Value of Checklists and the Danger of Scripts: What Legal Training Suggests for Testers,* he'll talk about the distinction between scripts and checklists, and how that distinction has helped him to approach many testing tasks in a way that provides structure but doesn't restrict exploration.

Rob Sabourin is the author of *I Am A Bug*. Cleverly disguised as a children's book, it's one of the wisest books on testing ever, and certainly the most concise. If you're a tester who wants to explain testing to anyone from three years old up to grandparenting age, buy them this book. Rob is one of the forces behind the peer Workshops on Performance and Reliability, and an extremely energetic and engaging speaker. With his wife Anne Sabourin, he'll be talking about *Applied Testing Lessons from Delivery Room Labor Triage.*

Tutorials will be given be **Jerry Weinberg, Hung Nguyen, Scott Barber, and Julian Harty**. I'm going to save stuff about that for another newsletter.

AST is a non-profit association, so the conference fees are rock-bottom even before the value is considered. We've negotiated a really reasonable rate for the hotel, too--$114 per night, which is practically unheard of for Toronto. (The facility, 89 Chestnut Street, is a University of Toronto residence while school is in session. While there will be free internet access for all conference participants, one quirk is that the guest rooms don't have televisions. There are TVs in common areas on each floor—and besides, everyone will be too busy testing, or talking about testing, and/or visiting the many fine pubs and restaurants in the area to worry about American Idol reruns.

There are three principal ways, for now, that you can support CAST.

1. We'd love to see you at the conference. You can register at http://www.cast2008.org/Registration.

2. We'd love your help in getting sponsorship for the conference. There's a complete sponsorship package available online at http://www.cast2008.org/Sponsorship.

3. If you can't make it, or you can't authorize a sponsorship, *please* send this information and these links along to someone who can. I've sent a brochure along with this newsletter, and you can find a copy at http://www.developsense.com/CASTBrochure2008-03-10.pdf.

What's A Tester To Do?

Recently I've been noticing a pattern of questions in the newsgroups, in personal correspondence, and in conversation. A tester is asking the questions, and the pattern goes like this:

1) I've got a product that's changing rapidly.
2) I have to test to make sure that the changes aren't breaking anything, plus I have to find new problems, if there are any.
3) There's some terrific problem that makes (2) slow or difficult or both. (It could be that builds come more often than the time perceived to be necessary for running a regression suite; the product is built using some weird technology that resists automation assistance; it's a big, complicated product, etc., etc.)
4) I have to decide whether to test everything, or perform some subset of my usual regression suite, and I'm worried that someone might be upset with my decision. (Or management is asking me "how long is it going to take me to test?") What should I do?

The answer lies in recognizing that the business, not the tester, is proper agency to answer the question "What should I do?" in the larger sense—that of the testing mission. Testers typically do not have control over the schedule, the budget, the scope of the projects, market conditions, contractual obligations, or hiring and firing and disciplinary authority. That is, *we don't run the project*. I've been hearing that for a long time from my mentors and colleagues—from Cem Kaner since 1996, from Jerry Weinberg and Johanna Rothman since 2002, and from James Bach throughout our working relationship.

I've lived on both sides of the "what should I do?" decision, too. As a project manager, I had sufficient authority and control over the project to slip the schedule. In that role, I depended on the testers to be my antennae. It was my decision to decide when the development work on the project was done sufficiently to ship the product. When the *development* work was done—*not* the *testing* work.

When I was a project manager, the development work was done when we believed that all of the problems that would prevent shipment had been fixed, and that we had no important unasked or unanswered questions remaining about the product, such that some unhappy answer would prevent shipment. Asking questions and answering them is what testing is all about. In that sense, the product doesn't ship until the testing work is done. The catch is that it's not the testers' questions that matter. The questions of the project manager matter. As a project manager, I depended on the testers to ask—and answer—more and better questions than I could ask and answer on my own, but they didn't make the decision on whether the product was ready or not. That was a business decision, not a technical one.

As a tester, *making that kind of decision is not my job*. Instead, it is my job to provide information to project managers so that they can make informed decisions about the project. As a tester, I need to have the confidence that I've given them the testing information that they need—the best information that I can, from as many dimensions as I can offer—but I also have to have the humility to recognize that my information isn't necessarily the most important information *to them*.

I never have to worry about how much time I need to test, because someone else is responsible for making schedule decisions based on the information they have. I'm happy to tell them, via an estimate, how long I think it will take to perform a *specific test activity*, but if I'm asked "how long will it take to test the product?", I'll pass and ask for a different question. I'll ask them what they want

to know about the product—and if they're short of ideas, or haven't recognized something interesting or important that I feel they *should* know, I have plenty of ideas on that. Some testers appear to get stuck at Step 4 above, when they might well be finished testing at Step 3, in which they have a problem that makes the product hard to test. Let's call that a *testability* problem, because that's what it is.

At any given point, testability problems may be the most serious problems in a product. Why? When a product is less testable, testing is slower. Things that slow down testing are likely to reduce test coverage, on the assumption that we'll run fewer tests and make fewer observations in the same amount of time. This means that a less testable product gives bugs more places and more time to hide.

A buggy program is less testable, and ironically, a product that has lots of bugs in it will take time away from test design and execution. A good test that doesn't find a problem tends to add to test coverage. A good test that *does* find a problem tends to require extra time for investigation and reporting, and this incurs opportunity cost. We can't cover some other part of the product while we're investigating and reporting. In addition, the bugs that we find may conceal other bugs, or may interact with other bugs to make troubleshooting more difficult. This is a powerful argument for developers writing cleaner code and testing the code that they write.

Here are some examples of Step 3 problems that I've heard of lately.

- My product won't be available for three more days, and management says that we'll have to ship at the end of this week. But I also have to run a full set of regression tests, and they take three days.

- My product doesn't keep a record of its actions, so I have to build logging into the test code.

- My product doesn't have an interface that can be addressed with a script (HTML with no element IDs, it's written in Oracle Forms, there are "security" measures in place such that we have to automate only at the front end…). Automation is going to be difficult and expensive, but management keeps insisting that we need to automate to save money.

- My product has to run on thirty different platforms, and management says that we have to test on all of them, but we only have to the end of next week.

- My product is hard to test because the installation program is so totally flaky.

- My product has tons of bugs in it, so we have to do lots of regression testing.

Interestingly, I rarely hear what either management or development are doing to do about this problems. And yet *these are manifestly problems for management or development to solve,* or at least to make some choices about.

- Why is a full regression suite perceived to be necessary? Perhaps because someone, somewhere believes that a change is sufficiently risky that a bunch of old tests will reveal a new problem. *Is a full regression suite the only way to address that risk?*

- Why doesn't the product have logging built in? Perhaps because someone, somewhere believes that developers have more important features to address, and that it's cheaper and easier to have

the testers develop the recording mechanisms. *When there is a perception that there are bugs to be found in the software, do we want testers seeking the bugs or writing logging code?*

- If management is adamant that testing be automated and scripted, shouldn't management direct that the product have a scriptable interface? Is there an interface that is scriptable below the GUI? *Does management recognize the cost of automation, especially when automation is hard? If it's impossible for the developers to provide element ID tags, will it be less expensive to ask testers to hack around that problem?* Oh, and one more: *Does management recognize the value of other kinds of testing?*

- Organizations often claim that a product will run on a given platform. Choices about that include testing on that platform; inferring support by testing on a similar platform; hoping for support, even testing isn't done; or changing the claim. *What are management's priorities in identifying and allocating testing resources to make the claim supportable?*

- When an installation program is flaky for testers who likely have substantial practice in working around installation problems, imagine what it will be like for customers, and for technical support. *Is management aware of the impact, not only on testing, but on other constituencies? Isn't an installable product a requirement, rather than a nice-to-have?*

- Regression is a problem, but it's not a problem that's *solved* by regression testing; it's a problem that is *identified* by regression testing. When previously fixed bugs come back, there's also ample room for new bugs to hide. What steps are being taken to trap regression problems earlier, so that less regression testing is necessary? What is being done to reduce the likelihood of side effects?

Most of these questions can't be answered effectively or reasonably by testers alone. Quite so: they're management questions, and testers should allow management to answer them. What should testers do? Testers should bring the news to management's attention, find out what the mission will be in the light of the new information, and *let management manage the project.* That's why they get the big bucks.

What I'm Up To

- I'll be teaching Rapid Software Testing to a corporate client the week of March 17, and to another the following week. Then it's off to Sweden for a week with yet another client.
- I'll be at the Software Test and Performance Conference in San Mateo, CA the week of April 14. See http://www.stpcon.com for details.
- I'll be at the first Kitchener Waterloo Software Quality Association software conference on April 23, giving my talk Why I Am Not (Yet) Certified . Check out http://www.kwsqa.org.
- I'll be at STAR East, giving a talk with Jonathan Kohl called "Angels and Devils of Software Testing" (guess who Jon plays; guess who I play). Look for conferences at www.sqe.com.
- I'll be in Sweden again the week of May 19, for a public version of Rapid Software Testing.
- I'll be in New Zealand and then in Australia for this year's STANZ conference in August.

Remember: CAST is July 14-16 in Toronto! Please come along!

You can always contact me at mb@michaelbolton.net. Keep in touch!