Blog: Of Testing Tours and Dashboards
From Michael Bolton at Developsense.
Original post at http://www.developsense.com/blog/2009/04/of-testing-tours-and-dashboards/

Back in the 1980s and 1990s, Quarterdeck Office Systems (later Quarterdeck Corporation)—a company for whom I worked—was in the business of creating multitasking and memory management products to extend and enhance to Microsoft's DOS operating system. The ideas that our programmers developed were so good and so useful that similar ideas were typically adopted by Microsoft and folded into DOS a year or so later. After each new version of the operating system, people would ask us "are you concerned about Microsoft putting more memory management stuff into DOS?" Quaterdeck's reply was always that, as long as Microsoft supported DOS, we would find ways to improve on memory management—and that we were delighted that Microsoft had legitimized the category.

I wasn't lucky enough to attend Dr. James Whittaker's presentation at EuroSTAR 2008, in which he described the concept of touring the software as a way of modeling and approaching exploratory testing. Fortunately, Dr. Whittaker has presented a number of these ideas as part of his recent Webinar "Five Ways to Revolutionize Your QA" on the UTest.com site, which came to my attention on April 1, 2009.

The touring metaphor in testing has been around for a while. I learned about it through James Bach's Rapid Software Testing course, which I started teaching in 2004, and of which I've been a co-author since 2006. In 2004—that's the first version for which I have my own copies of the course notes—Rapid Software Testing included several ideas for tours:

- **Documentation Tour:** Look in the online help or user manual and find some instructions about how to perform some interesting activity. Do those actions. Improvise from them.
- **Sample Data Tour**: Employ any sample data you can, and all that you can. The more complex the better.
- **Variability Tour**: Tour a product looking for anything that is variable and vary it. Vary it as far as possible, in every dimension possible. Exploring variations is part of the basic structure of my testing when I first encounter a product.
- **Complexity Tour**: Tour a product looking for the most complex features and data. Look for nooks & crowds where bugs can hide.
- **Continuous Use**: While testing, do not reset the system. Leave windows and files open. Let disk and memory usage mount. You're hoping that the system ties itself in knots over time.

But the idea had been around before that, too. Tours were also mentioned in the Black Box Software Testing course, co-authored by James and Cem Kaner, which I attended in 2003. They were part of a larger list of test ideas called "Quick Tests", which included other things like interruptions (starting activities and stopping them in the middle;

stopping them at awkward times; performing stoppages using cancel buttons, O/S level interrupts, ctrl-alt-delete or task manager, arranging for other programs to interrupt, such as screensavers or virus checkers; suspending an activity and returning later) and continuous use (while testing, avoiding the resetting of the system; leaving windows and files open; letting disk and memory usage mount, hoping that the system ties itself in knots over time).

Note that concept of touring wasn't terribly new in the BBST course notes and appendices either; skilled testers had been using them for a long while before . In 1995, Cem Kaner noted that the user manual is a test planning document; as he said in Liability for Defective Documentation, "It takes you on a tour of the entire program." Elisabeth Hendrickson gave a presentation at STAR East in 2001 called "Bug Hunting: Going on a Software Safari", which gave an overall list of test ideas using the metaphor of a tour. The idea of describing tours of a specific aspect or attribute of the product (namely the menu) appeared in an article by James Bach in the Test Practioner in 2002.

Much more serious work based on the concept of tours happened in 2005. Mike Kelly did some work with James Bach, and blogged some ideas about what they had discussed in an August 2005 blog post. Mike amplified upon that in his more complete list of tours (using the mnemonic FCC CUTS VIDS) in September 2005.

- **Feature tour**: Move through the application and get familiar with all the controls and features you come across.
- **Complexity tour**: Find the five most complex things about the application.
- **Claims tour**: Find all the information in the product that tells you what the product does.
- **Configuration tour**: Attempt to find all the ways you can change settings in the product in a way that the application retains those settings.
- **User tour**: Imagine five users for the product and the information they would want from the product or the major features they would be interested in.
- **Testability tour**: Find all the features you can use as testability features and/or identify tools you have available that you can use to help in your testing.
- **Scenario tour**: Imagine five realistic scenarios for how the users identified in the user tour would use this product.
- **Variability tour**: Look for things you can change in the application – and then you try to change them.
- **Interoperability tour**: What does this application interact with?
- **Data tour**: Identify the major data elements of the application.
- **Structure tour**: Find everything you can about what comprises the physical product (code, interfaces, hardware, files, etc…).

Dr. Whittaker does suggest some interesting notions of his own for tours in the UTest talk:

- **Money tour**: Test the features that users purchase the app for (which is rather like Mike's "user tour" above, I guess)
- **Rained-out tour**: Start and stop tasks, hit cancel, etc. (rather like Kaner's notion

of “interruptions” above)
- **Obsessive compulsive tour**: Perform tasks multiple times, perform tasks multiple times, perform tasks multiple times
- **Back alley tour**: Test the least-used features
- **All-nighter tour**: Keep the app open overnight (like Kaner’s notion of “continuous use” above)

In his related blog post, “The Touring Test”, Dr. Whittaker says, “At Microsoft a group of us test folk from around the division and around the company are experimenting with tour-guided testing.” Cool. He also says, at the top of the post, “I couldn’t resist the play on Alan Turing’s famous test when naming this testing metaphor.” The idea that he named tours independently is a little surprising, but when we think about the practice of skilled exploratory testing, the “touring” metaphor might be obvious enough to have been arrived at independently. These things happen.

For example, in 2005, James Bach showed me a bunch of test techniques that he called “grokking”. (Grokking is a word invented by Robert Heinlein that describes deep, meditative contemplation and comprehension.) I thought “grokking” wasn’t the right name for what James was describing, because the tests depended extremely rapid cognition and removing information, the very opposite of reflective contemplation. I was reading Malcolm Gladwell’s book *Blink* at the time, and I suggested that we label the techniques blink testing. Only later, when I was researching the history of similar observational approaches for an article I was writing on blink testing, did I find a reference to astronomer’s tool from the 1920s: it was called a Blink Comparator. It was fun to note that discovery, a little sheepishly, in the article. So I can understand how it’s easy for people to use the same label for an idea.

But then something else came up.

In the Webinar for uTest.com, Dr. Whittaker also presents the concept of a “low-tech testing dashboard”, in which he suggests using a whiteboard and coloured markers to report on project status. This suggestion isn’t just a variation on the Big Visible Charts that are recommended in the Agile literature; it’s strikingly similar to an idea presented James Bach at STAR East in 1999 in a talk called “A Low-Tech Testing Dashboard“, posted on his Web site since around that time, and also part of the Rapid Software Testing course (pages 136-146).

I am delighted that authors as well-respected as Dr. Whittaker and that companies as prominent as uTest and Microsoft are endorsing and helping to spread ideas on tours and dashboards. I think they’re worthwhile approaches, and I believe that such endorsement helps in the wider effort to get the ideas accepted. Yet I also believe that it would be a friendly and respectful gesture if Dr. Whittaker’s presentation included acknowledgement of prior work in field that it covers. It would be similarly helpful if books like Dr. Whittaker’s How To Break Software or Page, Johnson, and Rollison’s How We Test Software at Microsoft contained bibliographies so that we could more easily find references to some of the ideas presented.

What does the community think? How important is it to acknowledge earlier work?

This entry was posted on Thursday, April 2nd, 2009 at 2:41 pm and is filed under Uncategorized. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.