DevelopSense Newsletter *Volume 1, Number 2*

February, 2004

Michael Bolton
mb@developsense.com
http://www.developsense.com

Welcome

Please note: I've sent this newsletter to you either because you asked me for it explicitly, or because I genuinely thought that you would be interested in it. If neither is the case, please accept my apologies and let me know by clicking here, or send a message to remove@developsense.com.

On the other hand, if you like this newsletter, *please take a moment to forward it to friends or colleagues that you think might be interested*. If you'd like to get on the list, please click here, or send a message to addme@developsense.com.

Your email address is just between you and me. I won't give your email address to anyone else, nor will I use it for any purpose other than to send you the newsletter and to correspond directly with you.

Your comments and feedback are very important to me, and I'd love to share them with the rest of the recipients of the letter. Please send them on to me at feedback@developsense.com.

Getting Respect for Testers: Seven Tips

A recent discussion in the Agile Testing discussion group prompted some of these observations about how a tester can gain respect from developers.

The first observation is that, in the long run, although sponsorship is important, no person will make the case for you and your work more compellingly than you and your work will make it. A good manager will pave the way for you to gain credibility, but ultimately it is up to you. Things will happen if you make them happen, but if you sit idly and wait for respect, you'll still be waiting years from now.

I have found that interesting, committed people generally like to talk about themselves and what they do. Developers almost always appreciate testers who are genuinely interested in understanding the developers' work. The way to express your interest is to ask questions and to show a desire to learn what they're doing and how they're doing it. Start by engaging with a developer that is interesting and committed quality work, and is willing to talk with you. If there's a choice, choose the approachable ones first, but don't ignore the "unapproachable" ones. People with an "unapproachable" reputation may simply have been approached by the wrong people in the wrong way. Make yourself the right person, and approach them in *the right way for them*. That is: respect their terms. Learn about the Meyers-Briggs Type Inventory (MBTI) to help recognize, understand, and deal with the fact that people have different temperaments.

Second, reward their trust in you by making good use of the things they tell you. Use those things to help sharpen your own saw. Learn about the domain in which you're working, learn about the

programming language. Use the knowledge to devise powerful tests that will help to find problems before they're released. Try to encapsulate that knowledge for other testers, the support department, or the company's knowledge base.

Third, apropos of the last point, do your work in public. At one organization, I found myself consistently answering questions in company forums from less experienced techs and testers. My profile rose when I provided correct answers. Perhaps more surprisingly, my reputation didn't fall when the developers had to correct or clarify me, as long as I acknowledged the error and thanked them for the clarification. No matter what happened, the techs had their answer, I usually saved the developers some time, or I acted as a useful foil to them. I learned a lot in the process.

Fourth, don't waste their time. If they suggest that you read something for more information, do your best to read it; then ask them more questions about it, partly to clarify your understanding but also to signal your interest, your diligence, and your curiosity. Save time for developers by making bug reports and other communication as clear as you're able. Don't assume that you're doing a good job; an absence of feedback might mean that the developers need nothing more from you, but it might also mean that they're ignoring everything you send them. Solicit feedback to make sure that they're finding your reports effective: "Did you get everything you needed from that report? Is there another tack that I could be taking to make reports work better for you?"

Fifth, ask them how you can otherwise best help them to do their work. Consider ways to help them within the context of services that testers are supposed to offer in your organization. As an example, help the developer set up and maintain smoke testing systems. Ask to be invited to code reviews or inspections as a scribe. Be prepared to extend your department's services if you see a reasonable and justifiable need. If the boss hasn't given you permission, remember Grace Hopper's Law: it is easier to beg forgiveness than to receive permission.

Sixth, check your ego at the door. If you are excessively arrogant, no one will like you. On the other hand, if you behave timorously, as though you were a second-class citizen, people will find it easy to view you as one. Nobody emerged from the womb knowing his chosen trade; everyone learned this stuff from someone else. Seek explanations and understanding, and if you get attitude about that, don't take it personally. Behave like a peer and a colleague, and you'll more likely be treated as one.

Seventh, let your work speak for you. Excellent reports, compelling questions, and interesting tests will tend to attract positive attention. The best was to garner respect and support though? Just Find Cool Bugs. As I evolved in my career as a tester, I was consistently able, by skill and sometimes by luck, to find some really horrible problems. I've sometimes seen testers who seem reluctant to communicate bad bugs. Developers generally liked the nasty ones, because developers are fundamentally problem-solvers, and interesting bugs give them interesting problems to solve. The more interesting the problem we present, the greater our credibility. Therefore: Just Find Cool Bugs.

I'll be refining these points into a more presentation over the next little while; I would appreciate your reactions and feedback.

Tip of the Month: Mystery Keys and Windows

In my experience, few people—even dedicated keyboard users—have paid much attention to two keys that are standard on most keyboards. One is the Windows key, with an icon of the Windows logo on

it. Some people know that tapping the Windows key causes the Start menu and the Taskbar to appear. Few seem to know that the Windows key can also behave like a shift key, like Shift, Ctrl, and Alt. I did this once by accident, which led to an exploratory testing session in which I found a few useful combinations:

Windows-B causes the task***B***ar to appear so that you can see the applications that are running.
Windows-D minimizes all applications and displays the ***D***esktop
Windows-E causes Windows Explorer to appear.
Windows-F causes the Search (or ***F***ind) function to appear
Windows-L ***L***ocks the display and the keyboard to secure your computer when you leave your desk
Windows-M ***M***inimizes all applications and displays the desktop
Windows-R pops up the ***R***un command

The other Mystery Key is the one that looks like a little menu with an arrow on it. What does it do? Well, like a good exploratory tester, when I see a key, I press it. The Menu key (that is what I'll call it) displays the popup context menu for the part of the screen that currently has keyboard focus. (Serious exploratory testers will know that the Shift-F10 key does the same thing.) This is handy for folks like me who tend to be keyboard-centric. Think of the keyboard as a 101-button mouse.

Exploratory Surprise of the Month

Windows-U on my Windows XP system does something *completely* unexpected to me; it launches the Microsoft Utility Manager. The Utility Manager provides three tools

- a basic magnifier that uses the top portion of the screen to show an enlarged version of the area to which the mouse is pointing
- an on-screen keyboard, apparently for people who like to combine all of the disadvantages of the mouse and the keyboard
- the Microsoft Narrator, an extremely simple voice-synthesized screen reader. Unlike the previous two utilities, the Narrator is activated when the Utility Manager is launched. The result is that your computer suddenly starts speaking to you in a voice vaguely reminiscent of Stephen Hawking, reading off the attributes and contents of the foreground window. The most interesting thing about this, paradoxically, is that it doesn't work very well, but it sure is interesting to see—or rather hear—what it considers worthwhile to report to the user. As the utility itself notes, people with real accessibility needs will want to obtain a serious screen reader. However, Narrator is good for several minutes of amusement.

Tools of the Month: SysInternals

Mark Russinovich and Bryce Cogswell have been active in the Windows development and analysis community for quite a while. Back in the days of the SoftRAM debacle, they provided the analysis that demonstrated that SoftRAM—a utility that purportedly doubled the amount of available RAM on Windows systems—did nothing useful other than to relieve unsuspecting customers of their cash.

Since they, they have been developing and amassing a library of useful tools for testing and debugging. The Winternals site (http://www.winternals.com) contains their commercial tools—a recovery manager and other disk and system-level utilities—but the really interesting stuff for testers is at their

freeware site, http://www.sysinternals.com. You'll find Process Explorer, a very powerful tool which helps you identify which program has a particular file open and thereby assists in navigating through .DLL hell; FileMon, which allows you to track the opening, closing, reading, and writing of files; RegMon, which reports on programs' use of the Windows Registry; and lots of others.

I use these tools a lot to reveal information about what a program is up to. Process Explorer tells me about loaded .DLLs, including their version numbers and their dependencies. With the FileMon and RegMon tools, I've tracked down the cause of several mysterious bugs by observing that a program is trying to open the wrong registry key, or is attempting to read from a missing file.

The "Strings" utility (available in the Miscellaneous section of the site) finds text strings within a program. By default, it treats three consecutive characters as printable text. This is useful from two perspectives. First, it will allow you to see easily the more obvious typos and spelling errors in the program, even when you don't have a complete map of the product's menus and dialogs. Second, it will help alert you to the fact that your map may be insufficient—if you see text that isn't in your screenshots or prototypes, there may be some functionality in the program that hasn't been brought to your attention. Third, it will help you to note when strings intended for display to the user are stored within the body of the program, rather than in the resource table at the end—a no-no for internationalization and easy maintenance.

Online Resource of the Month

Cem Kaner hosts and moderates the Software Testing mailing list. Some of the brightest lights in the testing industry, especially those in the Context-Driven School, are regular contributors. Since the essence of Context-Driven Testing is skill and diversity, there are many different approaches and points of view expressed in the forum. Moreover, since it's a moderated group, discussion tends to be very focused and on-topic. Check out http://groups.yahoo.com/group/software-testing.

Shameless, Though Low-Key, Self-Promotion

I offer courses and consulting in software testing, quality assurance, and program management. For more information, please see my Web site at http://www.developsense.com, and my course brochure at http://www.developsense.com/SoftwareTestingInPlainEnglish.html.

On the Web site, I have recently revised my article on Pairwise Testing. See http://www.developsense.com/testing/PairwiseTesting.html

I'm going to be giving a public presentation on "Finding Bugs for Fun and Profit" to the Kitchener-Waterloo Software Quality Association for their April 28th lunchtime meeting. If you're in Kitchener or Toronto area and would like to attend, please drop me a line to let me know—for the Torontonians, I'll have up to three seats available if you'd like to hitch a ride.

That's it—see you next month!

---Michael B.