DevelopSense Newsletter *Volume 1, Number 1*

January, 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.

Getting Started…

This is a pretty strange business, this computer business. Three things came to me last month that got me riled.

Item: A friend of mine runs a small software company that makes a couple of genuinely useful products (I won't mention his name, or the name of the company or the products here for reasons that will become apparent). He was glad to have one of his products reviewed by an online newsletter whose brief is to evaluate commercial software for its readers. The author of the newsletter charges a substantial subscription fee, on the order of hundreds of dollars per year. In order to work properly, my friend's product must communicate with a host server, and must provide a serial number that is tailored to each registered user. That means that my friend can tell whether a given copy of the product has been run or not. He watched anxiously for the reviewer to register and run the product. The review appeared, and was very favourable. The only problem was that the reviewer never ran my friend's software—evinced by no connection was ever made to the server. Instead, the reviewer simply rewrote the marketing material that my friend provided, and posted that—in his fee-for-service newsletter!—as a finding of fact.

Item: Intuit Canada publishes a Canadian version of Quicken XG. One of the features touted on the box is the ability to download statements from bank accounts in the United States, in addition to Canadian ones. This feature was especially attractive to me, since I'm a dual citizen and I keep bank and credit card accounts in both Canada and the U.S.—so for the last couple of years, I've had to keep two sets of books and run two versions of Quicken—one for my U.S. accounts, and one for my Canadian accounts. The American version of Quicken supports dozens of U.S. banks, including the ones at which I have chequing and credit card accounts.

This spring I installed the new Canadian version of Quicken XG, I set up online banking and found a list of supported Canadian banks, but no American banks. I looked carefully through the product, and found that I simply could not obtain online access for any American bank. I called Intuit Canada to find out why. The explanation they offered was that the U.S. bank had to provide special support for requests from the Canadian version of Quicken. This is not a credible explanation. Most banks U.S. support Quicken, and most banks provide a Web-based online banking service. I can use both of those means to access my American accounts. The problem was simply that the Canadian version of the product doesn't provide technology to access to any U.S. banks. As of the end of November 2003, it is still impossible to find a single American bank listed in the updated version of the software. Although the product provides value to me, I believe that the claim on the box is false and misleading.

Item: the other day my mother called with a computer problem on her Windows XP machine. Internet Explorer was hanging with a protection violation as soon as she tried to start it up. This problem affected Windows Explorer and the Control Panel too—thus she couldn't see the file system, nor could she get into the Add/Remove Programs Control Panel applet. The exception message identified a dynamic-link library called VX2.DLL as being involved with the problem, but without access to the Net, I was unable to research the problem further. Fortunately, the command prompt was still available, so after a little remedial work (which I'll detail in a future message), I was able to get Explorer working. It turns out that VX2.DLL is a piece of spyware that reports on the machine's configuration and the user's Web surfing habits without giving anything like sufficient knowledge of its activities to the user. This is odious on its own, but when the software contains a defect that is causing hangs on machines all over the Internet...

The most remarkable thing about all this is that consumers by and large accept it. We've been trained to accept bad software by years of shoddy software and shabby marketers, aided by an ineffective press. Moreover, the software industry gets away with a lot by preserving the illusion that it's complicated, mysterious, and special. I don't believe that; I think that software and computer systems can be understood by almost anyone, as long as some provides decent explanations. But one problem is that we don't explain very well to our customers, or even to each other, what we do and how we do it.

So this newsletter is my tiny attempt to remedy some of that, at least in the field of testing and software quality. Accordingly, with this newsletter, I'll try to

- tell you about some test techniques that I use that might be obvious to some of you, but brand new to others;
- tell you about thoughts and experiences that I've had and resources that I've found in the pursuit of knowledge about software quality and software testing;
- tell you about software that I think might be useful to testers and others in the development process; and
- pass on the same kind of information from people on the list (that means you);
- tell you about what I'm generally up to;
- keep sending you more until you ask me to stop. If you want to be dropped from the list, please let me know. And,
- Apropos of the first item above, I won't charge a fee.

This should be fun.

Conference: Amplifying Your Effectiveness

In November of 2003, I attended the Amplifying Your Effectiveness (AYE) conference in Phoenix for the second year in a row. I recommend it highly.

The conference is centred on the work of Gerald Weinberg and his associates and students. If you're not familiar with Gerald Weinberg's work, I recommend that highly too. Those of you who have seen one of my presentations, taken one of my courses, or followed the work of the Context-Driven School of software testing, will probably recognize Jerry's definition of quality: "*quality is value to some person.*" This definition helps to explain why there are such strong differences of opinion about quality attributes; people—managers, marketers, developers, customers, and, yes, testers—have different criteria for assessing value.

Jerry started in the software business very early on; he was one of the programmers on the Mercury project that sent America's first astronauts into space. Over the years as a consultant, he came to recognize the premise that drives his work: *no matter how much it looks like a technology problem, it's always a people problem*. His books and teaching over the years built on this premise, incorporating influences from Virginia Satir, a family therapist, and the Meyers-Briggs Type Indicator (MBTI), which is a set of guidelines for understanding the differences between people's temperments. Jerry also hosts the SHAPE Forum, an online discussion forum that feels rather like a moderated group blog. (SHAPE stands for "Software as a Human Activity Practised Effectively"). Alas I missed the opportunity to attend Jerry's Problem Solving Leadership and Change Shop workshops; although still very active, he is moving towards retirement and has wound those down. AYE, however, continues.

It's an unusual kind of conference, oriented most strongly towards consultants and people who want to bring change to their organizations. Aside from independent consultants and teachers like me, there are also managers, programmers, testers, documenters—people involved at all levels of software development. Among other interesting features of the conference, PowerPoint presentations are banned! There are no overhead projectors, data displays, or handouts. The sessions tend to be conversational, or based on experiential learning. Seating tends to be in circles, rather than rows; attendees don't attend sessions so much as participate in them. The diversity of people attending and the depth of their experience and intelligence, combined with the mandate to exchange perspectives and ideas, make the sessions very valuable indeed. I'll tend to mention AYE and SHAPE in future articles.

The conference is held annually in Phoenix. The main conference lasts three days, Monday through Wednesday. On the Sunday before, there's a warmup day whose purpose is to explain the Satir System and the MBTI, and on the Thursday afterwards, there's also an extra day in which members of the SHAPE forum gather to meet and chat in person.

If you haven't read Jerry's work, I would recommend the Quality Software Management series. These books aren't cheap, but they're a valuable investment. His writing is engaging, conversational, and witty; humane rather than technological.

You can read more about the conference and meet some members of the community at http://www.ayeconference.com. You can find a good introduction to Jerry and his style in an interview, linked off the AYE page: http://bdn.borland.com/article/0,1410,30051,00.html.

Book Review: How to Break Software

Whittaker, James A., How to Break Software. Pearson Addison Wesley; Book and CD-ROM edition (May 9, 2002).

This is an important book about finding bugs, and thinking about finding bugs.

Most books that purport to be about testing are really about something else. They may be about planning, or process, or mathematics, or graph theory. Sometimes they're about making models of software to demonstrate that there are indeed jillions of paths through a given piece of software—hardly news to anyone who's bothered to think about it for a while. Sometimes they're about the underlying theory of the thing to be tested, such as "Web applications" or "security". All of these are useful things to think about, to be sure, and many books try to cover all of these topics to some degree. Few books cover the practice of finding bugs while testing to the extent that this one does, or in the same manner. Even fewer books provide such entertaining, cringeworthy examples of bugs in the field.

Successful bug-hunting depends on several things. It depends upon a desire to find bugs efficiently and effectively, and the belief that the bugs are there to be found in the software under test. That belief is founded on a "fault model", an understanding of how software works and how it can fail. Whether we have thought about it consciously or not, we all have fault models of some sort. One common fault model is "if the specification says X, the software will do something that is not X". The weakness with this fault model is that the specification might not say X; the specification might not even be available at all. What's a tester to do in that circumstance?

Well… test. I don't need a great specification—or any written specification at all—to start testing and finding bugs effectively. As Brian Marick once said, “most specifications are useful fictions”. A specification is just one map of the software that I’ve been given to explore, but an explorer doesn’t depend exclusively on maps. I have some understanding of how the operating system works, whatever that operating system might be. I have an understanding of how user interfaces generally work, especially in the Windows and Web worlds. I know something about programming languages, and some of the more common pitfalls that they introduce. I also know something about the way programs and programmers can screw up—that is, I know something about certain risks.

So when I’m testing, even if I have a written specification, I'm also dealing with what James Bach has called "implied specifications" and what other people sometimes call "reasonable expectations". Those expectations inform the work of any tester. As a real tester in the real world, sometimes the things I know and the program are all I have to work with. How to Break Software inspired me to think again about the ways that I approach testing a piece of software, irrespective of the amount of supplementary material that I have.

Good models help people to understand a subject, and good fault models help testers to find bugs. In How to Break Software, Dr. Whittaker presents both. He begins by positing two elements with which a piece of software interacts: the human and the system. Software interacts with humans via a user interface; software also interacts with the file system, the operating system, and other software. Each of these interfaces provides opportunities for finding errors; Dr. Whittaker calls those opportunities "attacks", and he provides a smattering of them for each interface. Moreover, and more importantly, for each attack he provides advice on when to use the attack, the kinds of faults that are vulnerable to

the attack (that is, the theory of error), how to determine the success of the attack, and how to conduct the attack. It should not be hard to imagine risks in the kinds of software you test, and use a similar structure to prepare your own taxonomy. Thus, as I am both a tester and a trainer, the key benefit of the book to me is that it provides useful ways to think systematically about exploratory testing.

The book is easy to read, too—it's written in a relaxed, conversational style. I found that, by using (mostly) Microsoft Office products as a way to demonstrate the attacks and find bugs, Dr. Whittaker appealed to my sense of schadenfreude (the malicious delight in the misfortunes of others). As Jerry Weinberg says, the goal is to avoid learning from your own mistakes; instead, the idea is to learn from the mistakes of other people. Picking on Office is, of course, a little like shooting fish in a barrel. Microsoft products are famously buggy, but they're also simply famous, and they're widely available. Although the book was published in 2003, the examples are mostly from the Microsoft Office 2000 suite. Office has, of course, been updated since then, but more people have the Office 2000 than Office 2003, and the older products aren't being updated, which gives the book a slightly longer shelf life. I found myself giddily checking to see if the bugs were still there, and sure enough, they were. Fun is important. Dr. Whittaker getting together with other testers and talking about bugs for fun. Good point—I believe that people learn more easily when they're talking to each other and having fun.

The book also includes a CD, which contains two programs, Canned HEAT and Holodeck. The former program's name is an acronym of "Hostile Environment Application Tester"; it's designed to help simulate various kinds of faults such as a network disconnection, memory and disk shortages, and so on. Holodeck contains the fault-injection functionality of Canned HEAT, and adds logging capabilities. I've given these programs only a minimal look, so I'll reserve comment for the time being.

The book has been positively reviewed by a number of people on Amazon.com, but I would like to conclude this review by responding to the negative reviews there, because I think those reviews are instructive about some prevailing nonsense in the testing world.

One reviewer complains that "The techniques that were taught in this book can be easily summarized in three words: 'Boundary Conditions Checking'". An equally valid review would summarize Moby Dick in three words as "whale hunting boat". The problem here is that the reviewer has apparently missed all of the material that *isn't* boundary checking; moreover, he has missed that the point of the book is how to think systematically about fault models and exploration.

A second reviewer claims, in a review entitled "Another Rationalization for Flailing (Hack Testing)", "The book is basically a rationalization for lack of planning, lousy requirements updating, and an endorsement for undisplined and unstructured software development." This is, in my view, also incorrect: the book makes contains neither rationalizations nor endorsements. How would this reviewer test third-party commercial software for compatibility with his application? If he were wise, he would use some of the tests here, and would think systematically—probably in the way that the book prescribes, or something like it—about other tests of his own invention.

A third writer titles his review "Not for J2EE Testing", and he asserts "I wouldn't recommend it to anyone testing REAL software. I would recommend it to all Fat Client testers using MS products." On the tenuous assumption that Java 2 Enterprise Edition is The Only Form of REAL Software, I would challenge this reviewer to show me some of his software, and I'll find bugs using exactly the kinds of methods outlined in this book. Trust me.

It’s true that some of the information presented in How to Test Software is quite basic. Mind, as a tester, testing trainer, and user of software, I've seen a lot of software—a *lot* of software—with some pretty basic bugs. Mission to Mars, anyone?

A good book should help and inspire you to think for yourself. If your development culture has closed your mind to extending the ideas in a book, you probably won't like it much—but then you probably won't be able to function very well when you move to a different culture. That is (sad to say) you won't be a very good tester when you leave your cocoon. In fact, if your mind is closed, you're probably not a very good tester now.

Does this book tell you everything you need to know about testing? No, of course not. Why should it? How could it? Does any other? On the other hand, I believe that this book is very useful if you keep your mind open, accept its lessons and examples, and apply them to your own projects, your own environment, and your own thinking. Open your mind, and jiggle your fault model. We need more testing books like this one.

You can order the book from Amazon by clicking here.

Postscript: While looking for the bibliographic information for this book, I found a bug in the display of the page. Note the last line below.

see larger photo

List Price: ~~$35.00~~

Price: **$28.41** & This item ships for **FREE with Super Saver Shipping**. See details.

You Save: $6.59 (19%)

Availability: Usually ships within 24 hours

Want it delivered Monday, February 2? Order it in the next 0 hours and 0 minutes, and choose **One-Day Shipping** at checkout.

Web Resource: StickyMinds.com

StickyMinds is a Web site that is associated with Better Software magazine (formerly Software Testing and Quality Engineering, or STQE: “sticky”—get it?). You’ll find a good deal of interesting and useful information there. There’s a fair deal dross, too, but it’s worth a regular visit and a search when you have a specific topic in mind.

http://www.stickyminds.com

Where I'll Be Later…

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.

Concluding Remarks

Most of you are probably aware that I have a Web site at http://www.developsense.com. Over the years I've been building a library of essays, reviews, and resources. I'll try to keep you informaed of updates via this newsletter. Please drop in every now and again.

As always, I'm very interested in your comments, and I'm happy to circulate them if you like. Please let me know if you'd like to make them publicly or privately.

So long for this month,

---Michael B.