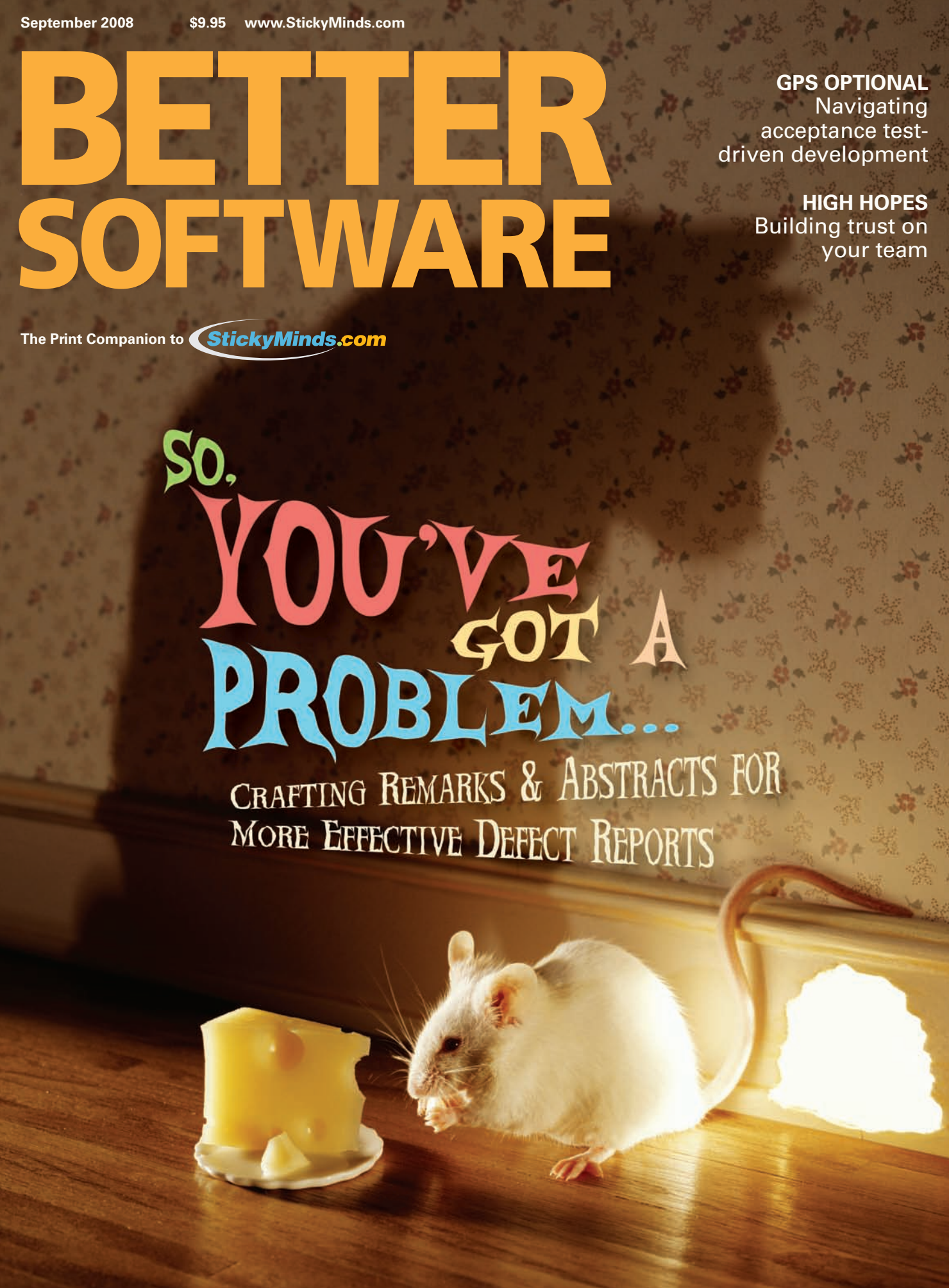
September 2008 $9.95 www.StickyMinds.com

# BETTER SOFTWARE

**GPS OPTIONAL**
Navigating acceptance test-driven development

**HIGH HOPES**
Building trust on your team

The Print Companion to StickyMinds.com



# It's in the Way that You Use It

by Michael Bolton

ISTOCKPHOTO

People sometimes ask me what my favorite test tool is, and my answer often confuses them. They seem to expect me to name a program that automates the operation of an application, makes some process more repeatable, or allows a machine to drive as quickly as possible through an application. But if you think driving is merely about getting from one point to another, ask a New York cabbie.

Rapid testers don't think of test automation merely as something that controls a program and checks for some expected result. Instead, we think of test automation as *any use of tools to support testing*. So my favorite test tool changes from moment to moment, but it's often (drum roll) my text editor. For years, I've used an editor called TextPad (see the Sticky Notes for a link). I'm not claiming that it's the best editor available, but it's *my* favorite. There are plenty of other text editors, and whichever one you use, if you're happy with it, I am, too. So what do I use a text editor for?

**Writing**. A lot of my work as a tester involves creating and editing text—taking notes, editing text, creating Fitnesse tables, building Web pages, or writing articles like this one. With my text editor, I can do these things efficiently without extra overhead like the tables, formatting and layout tools, styles, and thesauruses in sophisticated word processors. These extra features sometimes distract me or entice me to use them needlessly. As Marshall McLuhan said, "We shape our tools, and thereafter our tools shape us [1]." I sometimes use a text editor to emphasize to myself that I'm working on something temporary or preliminary. That's not to say that a text file is never important; a commercial software company that I worked for used to track bugs and feature suggestions by using a plain old ASCII text document. For us, the cost and overhead of an elaborate test management tool wasn't worth it when ideas could be captured concisely, compactly, and sufficiently in text.

**Programming**. Good text editors support programming by providing syntax highlighting, which recognizes the programming language that I'm using and changes the color or font of keywords, symbols, and variables to make them more distinguishable. This improves readability and provides some cognitive hints to alert me when I'm making a mistake. More sophisticated programming environments automatically add closing parentheses, braces, or brackets when I type the opening symbol; provide context-sensitive help for the languages and libraries that I'm using; and offer very sophisticated ways to map and model larger programs. When I'm working on a large project, I find these features essential, but for a quick-and-dirty throwaway script, a good text editor is just fine.

**Cutting and pasting**. Unlike Notepad, TextPad can mark blocks of text in columns, which allows me to open a file that's organized as a table, grab specific blocks in columns, and manipulate them or copy them to another program. This is often handy in combination with command line redirection to supply input from a file or output from a program to a text file. In Windows, it's also possible to extend the virtual size of the command window, copy output from programs, and paste it into a text editor for processing.

**Data generation**. I often use a testing technique called an input constraint attack—hurling huge amounts of text at an input field to see whether the program protects itself from buffer overflows that could compromise security or stability. To generate data quickly, I type ten characters and then hit Ctrl-A (select all), Ctrl-C (copy to the clipboard), and then Ctrl-V ten times (paste ten copies). Now I have a file of one hundred characters. Then I hit Ctrl-A, Ctrl-C, and Ctrl V ten times again; now I have a thousand characters. Repeat that cycle a few times with a final Ctrl-A, Ctrl-C, and Ctrl-V, and I have a million or ten million characters on the clipboard, suitable for pasting into an input field, an application, or a Web page. Try it! The results are often fascinating—data corruption, strange behavior, or a crash. Danny Faught and James Bach's PerlClip tool [2] has more features for this task, but a text editor will do many useful things in a pinch.

**Searching and replacing**. Sometimes I'm looking for words, strings of words, or patterns of characters inside log files, Web pages, XML pages, comma-delimited files, or binary files. Sometimes I want to replace strings of text with something else, or delete markup

so that I can focus on the text. To assist with these tasks, better text editors include support for regular expressions—powerful forms of notation for seeking, matching, extracting, or replacing patterns in text. See Jeff Friedl's *Mastering Regular Expressions* [3].

**Learning.** The Perl, Python, and Ruby scripting languages provide powerful support for regular expressions, but it can be hard to appreciate their value until you've learned to use them well enough to perform your first miracle. More complex regular expressions can be hard to learn without visual feedback and recoverability. Using the text editor to step through pattern matching and substitution, seeing changes in the text in real time, and being able to press the Undo key helped to accelerate my learning.

**Data inspection.** When I need to learn something about a file—its content or format—I often start investigating by opening it in a text editor. I might see a header that gives me clues, data in clear text, or repeated patterns of strings or blanks. I often scroll through a log file very quickly, using a defocused perspective to identify patterns of output or dramatic differences from the norm. (We call this "blink testing" or using a "blink oracle." See the StickyNotes for more information.) I can also use the editor to delete or change characters as a quick way to modify a clean data file to see how a program handles file corruption.

My colleagues and I sometimes use our editors for things that their publishers might not have expected. When I try to pour a lot of data into an input field, and the input field truncates the excess, I copy the field and paste it into the editor to get a word or character count right away. I often use text files as a backup mechanism when I'm entering text into a Web page. Jon Bach says that he uses text files as longer-term, safer storage for the clipboard. We both use text editors to peel the formatting off rich text when we're pasting large volumes of text to disjointed cells in an Excel file. James Bach once programmed his text editor to remind him to take testing notes every few minutes. When I forget the syntax for a tag or a special character in HTML, or the character associated with a particular ASCII code, my text editor is the handiest and fastest reference.

There are several points to all this. First, the simpler the tool, the more flexible, adaptable, and useful it might be. Think of the difference between a $50 Swiss Army knife and a $15,000 computer-controlled table saw. Second, like "quality," "purpose" is not something inherent in a product. Purpose is a relationship [4] between the product and the person who is putting it to some use. Third, some of those purposes are intended by the designers and adopted by the users, but novel and surprising purposes may emerge as we use a product—or as we test it.

As testers, we're usually asked to defend the value of a product by finding important problems with respect to its intended purpose. However, if we think of our products as tools that different people will use in different ways, we may *add* value by discovering purposes that haven't yet been recognized. For example, if we think of a test tool as "any tool that supports testing," even the humble text editor might become a powerful test tool. **{end}**


References:

1] McLuhan, Marshall. *Understanding Media.* The MIT Press,1994.

2] www.satisfice.com/tools/perlclip.zip

3] Friedl, Jeffrey. *Mastering Regular Expressions.* O'Reilly Media, Inc., 2006.

4] Weinberg, Gerald M. *Introduction to General Systems Thinking.* Dorset House Publishing Company, Inc., 2001.


**What tools do you use that you or others might not have recognized as a test tool?**

Follow the link on the StickyMinds.com homepage to join the conversation.

## Sticky Notes

**For more on the following topics go to www.StickyMinds.com/bettersoftware.**

- TextPad
- Blink testing