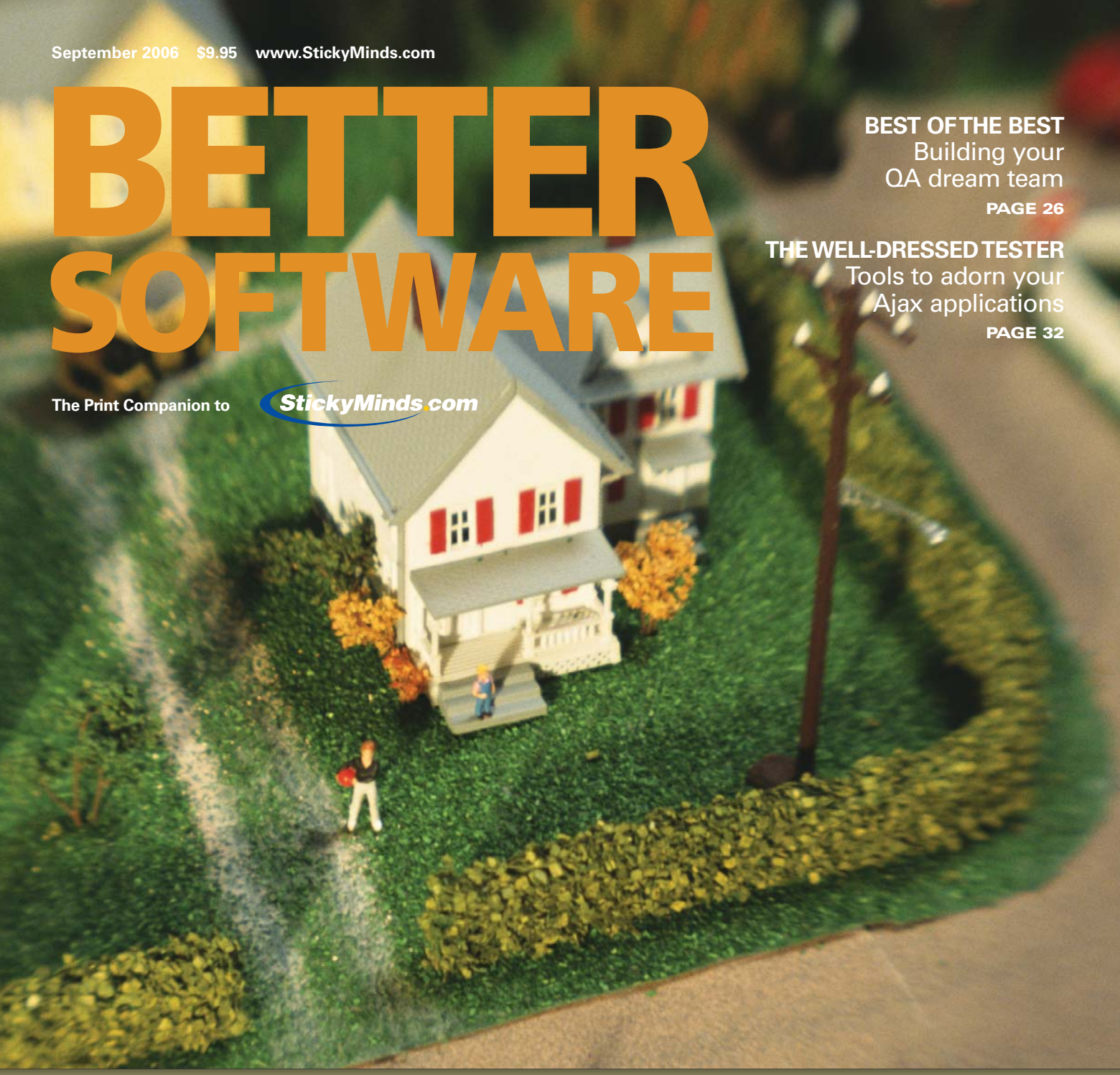
September 2006 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

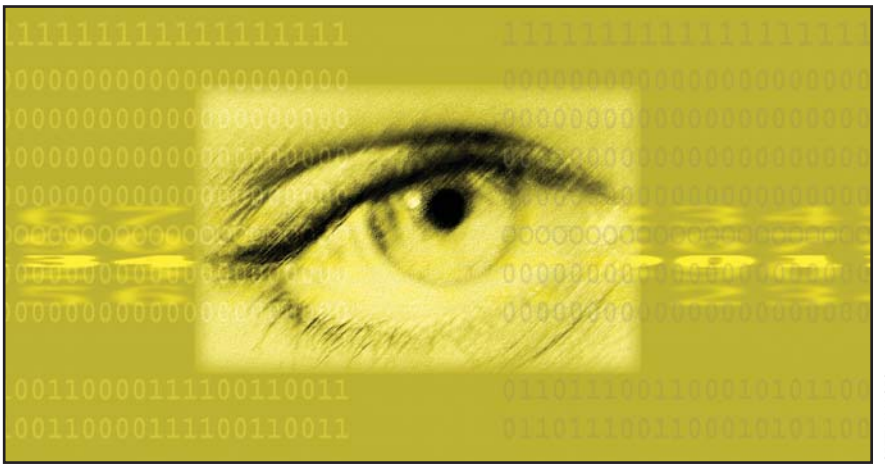


CODE Improvement

Create Designs With Curb Appeal

Blink . . . or You'll Miss It

by Michael Bolton



GETTY IMAGES

For several years, Malcolm Gladwell has been one of my favorite writers at the *New Yorker* magazine. At the beginning of 2005, he released his second book, *Blink: The Power of Thinking Without Thinking* (see the StickyNotes for more information). He says that the book is about "rapid cognition, about the kind of thinking that happens in a blink of an eye." It became a best seller but was criticized harshly by some because, to those reviewers, it seemed to suggest that rapid cognition was better than deeper, more reflective thought. On his book tour, Gladwell responded to those criticisms by summarizing the book in these four points:

1. Snap judgments are a central part of our decision-making process.
2. Snap judgments are vulnerable to corruption by forces that we're not aware of and don't understand.
3. We can sometimes improve the quality of our snap judgments by removing information.
4. Instead of changing the decision maker, change the context in which the decision is made.

This reminded me of some approaches to testing and observation. My colleague James Bach once showed me a rapid testing technique that he had caught himself using. While investigating a hard-to-reproduce problem, he had taken a log file from a Web server and had pasted each line into a cell in Microsoft Excel. He used Excel's conditional formatting feature to change the color of cells that contained a particular error message. He then zoomed out to a 25 percent view, making the actual data invisible, but we could still see regular, repeating bands of color. When we noticed irregularities, zooming back in allowed us to see what they were. The pattern always varied just after a connection had been made to one particular server. Had we looked at the data carefully, at full size in black and white, it would have been much more difficult to see the exceptions. We had changed the context in which we observed the data, and we saw the problems more easily by removing information.

Another colleague, Earl Everett, showed James and me a toy program that he had written for teaching testers. The program adds pairs of positive or negative integers of one or two digits. You can enter your own numbers or press a button to have the program place random integers into the input fields. James held down the Enter key over the button that generated the random numbers. Numbers flashed by in both fields faster than we could identify them, like the hundredths and thousandths digits on a race clock. Almost immediately, we saw two bugs: one bug occasionally pops a three-digit number into one of the input fields; the other prevents negative numbers from appearing in one field. The numbers flashed by too quickly to see, but the missing flicker of the minus sign in one field and the occasional third digit in the other were obvious. I have no idea how long it might have taken for me to identify those patterns had I logged single presses of the Random button.

We named these "blink tests." A blink test is any test that alters some way of viewing the application—size, time, position, focus, or sensory mode—to take advantage of people's ability to spot patterns (or pattern exceptions) quickly. Like snap judgment, rapid pattern recognition is something that testers use all the time. It's not always reliable, it's often improved by removing information, and it can be improved by changing the observer's context.

I've been using blink tests for my entire testing career. When we soften focus and flip quickly between two screens that are supposed to be identical, any difference between the screens shows up as a noticeable flicker. Recently, I used a blink test to spot differences between the ways that two browsers formatted certain Web pages. I loaded the same page under each browser, used Windows' Alt-Tab key combination to flip between the two, and defocused. I found inconsistencies in the handling of cascading style sheets and prevented bugs that would have been blamed on our application.

James later sent me another blink test—some video that he had taken of his fish tank, but in time-lapse photography. Several hours went by in a couple of minutes. Suddenly we could see all kinds of behavior patterns that would have happened too slowly to notice in real time.

We didn't invent blink tests, but we put a new name on something testers have always done. There are a lot of precedents for blink tests outside software testing, too. Edward Tufte, in his outstanding course on presentations, shows how to use repeating patterns in charts, graphs, maps, and observers'

notebooks (see the StickyNotes for a link to Tufte's Web site). He also introduces tiny word-size graphs he calls "sparklines" to illustrate important trends and data points. Ward Cunningham uses a wonderful approach that he calls "signature surveys" to spot patterns in source code by removing the code and leaving only the punctuation.

Blink tests don't have to be visual. As anyone who watches medical dramas on TV knows, irregular beeps from monitoring devices—or applications—can alert us to problems.

Blink tests aren't new. In 1872, Leland Stanford wanted to determine whether all four hooves of the horse left the ground at the same time during a full gallop. Eadweard Muybridge, a photographer Stanford hired to solve the problem, realized that a series of photographs a short time apart would capture the moment, and by 1878 he had arranged the experiment. Muybridge set up fifty cameras, whose shutters were released by tripwires that were triggered by the horse's hooves. The answer was not only clear, but surprising: all four feet were off the ground while they were folded under the horse. (Conventional wisdom said that all four feet were off the ground when they were at maximum extension forward and backward.)

Some of us do blink testing naturally. My eye immediately jumps to the typographical error on the restaurant menu. (Some people can multiply five-digit numbers in their heads. I find menu typos.) I tend to see similar problems in applications and source code listings, while other testers see other problems more easily than I can. If you have someone on your project who sees things differently than you, that's likely a good thing.

Training yourself to see things differently is also good. Some people seem to get better at rapid observation and evaluation with practice, especially after they've been fooled by one instance of a problem. We've been using examples from the *Surprising Studies of Visual Awareness* DVD, produced by Daniel J. Simons at the Visual Cognition Lab at the University of Illinois (see the StickyNotes for a link).

Though we can improve our perceptual skills, we're all limited to some degree. Compensate by using tools to show some kinds of information and filter others. As Gladwell says, "There are good ways of fixing the individual decision maker . . . We can find ways of editing out non-essential information."

The trouble is, when we're testing, we don't always know which information is essential. In order to find out, we must look at our products from many angles. General systems theory—a critical component of testing skill—says we leave out enormous amounts of information when we model, to pay better attention to that which really interests us. That's easier when we're in a confirmatory mode. Blink testing, by contrast, tends to be exploratory; it's about removing information deliberately and aggressively to see otherwise invisible patterns, without knowing in advance what we'll find. Blink tests are intended to supplement—not replace—more thorough forms of testing and observation.

Just to show that nothing in software testing is really new—not even the names we choose—while researching this column, I found that the McCormick Museum has in its collection an apparently homemade instrument that allows astronomers to alternate instantly between views of two photographic plates (see the StickyNotes for a link). With this device and others like it, astronomers found previously unknown variable stars, comets, and planets (including Pluto). The device is called a blink comparator. **{end}**

Michael Bolton lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries as part of James Bach's Rapid Software Testing course. Michael is also program chair for the Toronto Association of System and Software Quality. He is a regular contributor to Better Software *magazine. Contact Michael at mb@developsense.com.*

Sticky Notes

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

- *Blink: The Power of Thinking Without Thinking*
- Edward Tufte's Web site
- The Visual Cognition Web site
- The blink comparator

What ideas do you have for quick tests and quick observations?

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