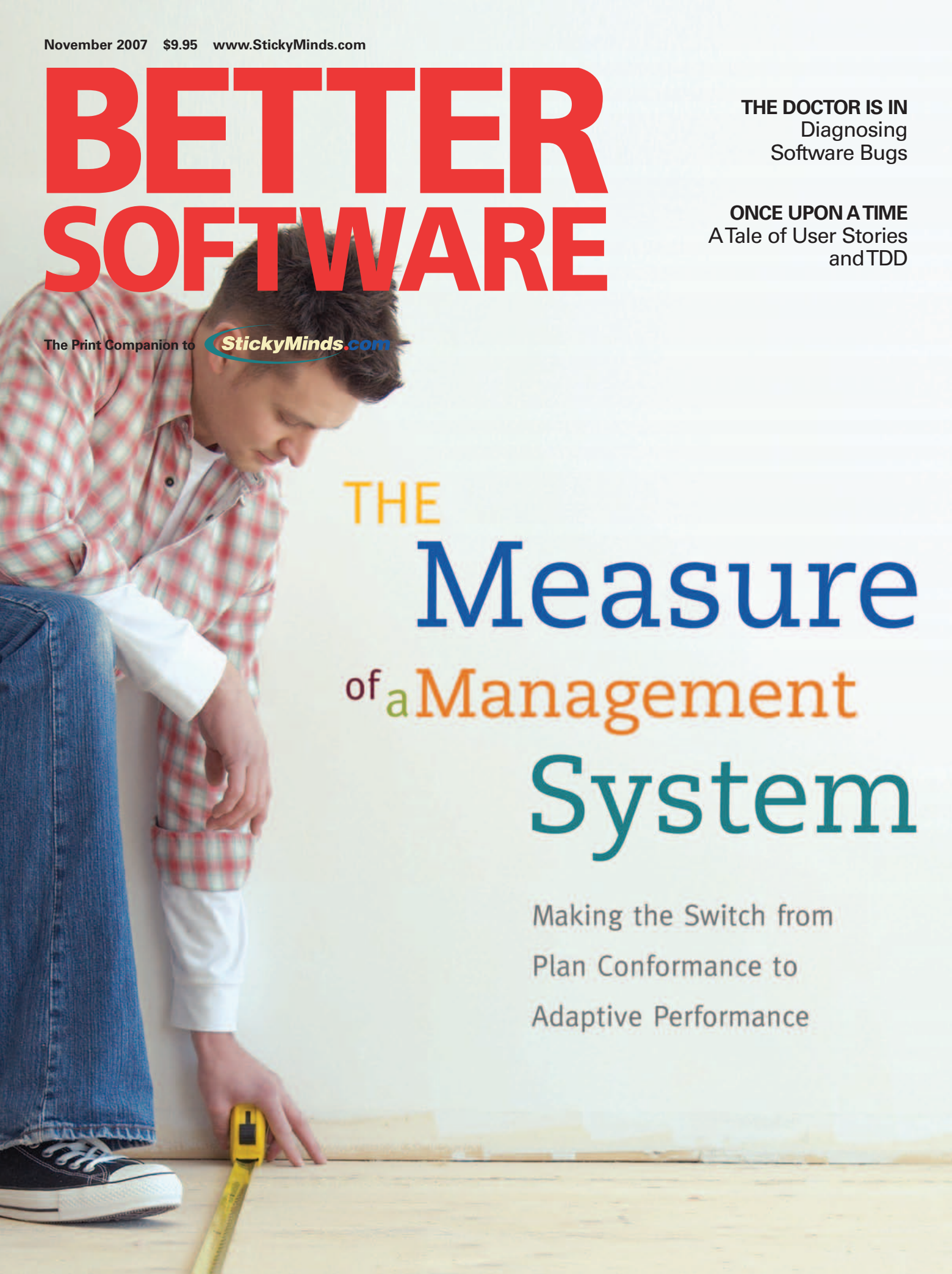
November 2007 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

THE DOCTOR IS IN
Diagnosing Software Bugs

ONCE UPON A TIME
A Tale of User Stories and TDD



THE Measure of a Management System

Making the Switch from Plan Conformance to Adaptive Performance

How Testers Think

by Michael Bolton

ISTOCKPHOTO

People think in models and metaphors; we seem to be hard-wired for them. Metaphors help us make sense of the world and deal with new things. Throughout the years, people have applied various metaphors to software development, including the notion of software development as an engineering discipline. Traditionally, engineering involved tangible, physical things, but software has enough in common with machines and development has enough in common with construction that the metaphor has served us well. When thinking of how to improve our products and our processes, we ask "What would engineers do?"

Metaphors are heuristics—fallible methods for solving problems and for learning. The engineering metaphor has become so pervasive that we're beginning to take it literally rather than metaphorically. We still have things to learn from engineering, but there are other fields from which we can take useful approaches, too.

This idea struck me as I began reading Dr. Jerome Groopman's *How Doctors Think* and recognized that many of the issues doctors face in diagnosing diseases are similar to those we face as software testers. Doctors work under conditions of extreme time pressure; they must adapt their choices to their clients and contexts, which may change at any time. They must identify and investigate symptoms of trouble while recognizing that they cannot know everything there is to know about the system. Moreover, doctors are beginning to recognize that complex systems—namely human patients—are resistant to discrete, linear, decision-making processes. Instead, doctors often make decisions based on rapid observation, instant analysis, and incomplete information—that is, by heuristic processes. In addition, medicine is a pursuit in which technical aspects are often overemphasized at the expense of human values.

On several occasions in Groopman's book, a doctor is presented with a difficult diagnostic problem. In one case, a woman in her early twenties, complaining of intense pain to the point of vomiting after meals, presents herself to doctors. They diagnose her with anorexia and bulimia and later suggest irritable bowel syndrome. None of the treatments offered by the doctors provides long-term relief. After fifteen years of steadily deteriorating health and weight loss, she is referred to a doctor with a reputation for solving tough problems.

He asks her to tell her story, in her own words, from beginning to end. He listens calmly and sympathetically with an open mind, observes keenly, waits for the patient to finish her story, and then asks her a lot of questions. After performing tests and collecting more data, he recognizes something the other doctors had missed—that the patient might have a physical ailment in addition to a psychological one—and diagnoses celiac disease, an autoimmune disorder. Subsequent tests and a rapidly recovering patient confirm that the diagnosis was correct. Asked how he did it, the doctor replied that he continually asks himself two questions: "What else could this be?" and "Is this the worst problem this patient could have?"

Several years ago, I worked on a product that was closely tied to an operating system. One of our test systems had been cobbled together from spare parts around the test lab, and this particular machine wouldn't boot properly when our product was present. One of our best developers (let's call him Ron) thought about the problem on and off for three weeks. He became convinced that it was just a flaky machine—there was some incompatibility between its components that our software exposed, which didn't represent an underlying problem that our code could address.

Our testers were convinced there was more to the story. Another developer (and expert tester), Pat, was willing to believe the testers and thought through what happened at boot time. He modeled the system into a set of components and the boot process into a series of tasks, noting the interactions between our software and the other elements of the system. He theorized that the hard drive controller might be involved in the problem, set up the machine under the debugger, and ran some tests. In a few minutes, he found that when our software was present, the drive controller didn't initialize properly under a certain set of easily reproducible conditions. When the testers tried our product on a computer that had been in use on another project—a machine that had the same

controller but a brand-name system—it crashed on boot. Pat threw in a short routine to special-case this drive controller, he squashed the bug, and we shipped on time. How had Pat solved the problem so quickly when Ron couldn't?

Ron had suffered from several significant, related cognitive errors, all of which have parallels in *How Doctors Think*. The bias that allowed Ron to ignore the problem was *fundamental attribution error*—explaining something entirely by a preconception or a stereotype ("She's anorexic"; "Machines that have been assembled from spare parts are intrinsically unreliable"). He attributed the problem to a *single path of causation* ("bulimia"; "a flaky controller") rather than interaction between multiple parts of the system. *Anchoring bias*—dropping anchor on an idea and not moving from it—allowed him to get stuck. (The doctors were stuck for fifteen years; the developer for three weeks.)

Like the successful doctor, Pat used applied epistemology combined with empiricism. "If I believe something," he told me later, "I ask myself *why* I believe it, and then I test for it." He pulled up the intellectual and emotional anchor and tried to abandon preconceptions that would get in the way of solving the problem. He kept his mind open to possibilities; there might be a bug in our code, in the disk controller, or in the interaction between them. He noted that the controller worked with products similar to ours, which allowed him to narrow down the conflict to one of our product's unique features.

As testers, we never know for sure the deep truth about any observation, so excellent testing includes open-mindedness and critical thinking at every stage of the testing process. We think we know something, but it's only what we know so far. When tests aren't revealing important new information, we might decide to stop testing, but we also might consider that we're using insufficient tests or making insufficient observations. Continued questioning is central to good testing. Contrary to much of our profession's folklore, a good testing question doesn't necessarily have a definitive, expected result. Tests that are designed to confirm a prevailing theory tend not to reveal new information. Collaboration and consultation tend to be more powerful than working alone—especially when we're stuck. If you're in an organization that assigns work to a single tester per feature, consider paired testing sessions with another tester. Note the flow of ideas, your progress—and the bugs you find.

People learn from stories. In software testing, we compose, edit, justify, and narrate two kinds of stories. We tell the *product story*, which is everything that matters about how the product can work, how it has failed in tests, and how it might fail in the field. We also tell the *testing story*, which is about how we have configured, operated, and observed the product; the things we haven't tested yet; the things we won't be able to test at all; and why we believe that what we've done is good enough.

Medicine is a metaphor, and metaphors get their power from the combination of comparison and contrast. There are a lot of ways in which testers *aren't* like doctors—and programs aren't like patients. For one thing, patients can tell their stories; software has to be run so we can observe it. At a certain level, software can be considerably more deterministic and easier to diagnose than human patients. As testers, we're rarely under the same kind of time pressure and emotional burden as doctors. Still, learning how doctors think may teach us some important lessons about how testers *could* think. **{end}**

Michael Bolton lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries. He is co-author, with James Bach, of Rapid Software Testing *and a regular contributor to* Better Software *magazine. Contact Michael at mb@developsense.com.*

From what other professions do you take testing lessons?

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