July/August 2005 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

WE'RE ALL IN THIS TOGETHER
Create Agile subteams that work for the greater good

PAGE 34

THE NOT-SO-GREAT DIVIDE
Factors to help you span the gap between disciplines

PAGE 12

The Cream of the Crop

PAGE 28

Milking Reuse for all It's Worth

Mission Critical: Visualize, Personalize, Humanize

by Michael Bolton

In the last column, I introduced Rapid Testing, a skills-based approach to software testing. For the next few columns, we're going to look at one of the key skills of Rapid Testing: critical thinking. Testers provide a service to management in the form of information, which allows management to make informed business decisions about software. As testers, we provide better information—and make better decisions possible—when we think critically about software.

What does it mean to think critically? The purpose of critical thinking is to make reasoned, dispassionate, thorough, and accurate assessments or evaluations. These evaluations are based on detailed observation, diligent collection and weighing of evidence, recognition of significant similarities and differences, awareness of bias and blind spots (and to the greatest extent possible, the elimination of them), and continuously reapplying what we've learned. Critical thinking requires us to place the object of our thinking in context. It also requires us to examine, question, and improve the ways in which we ourselves are thinking and observing.

In meetings, specifications, and hallway conversations, most software projects with which I've been involved have referred to "the user"—an abstract, undifferentiated term by which we identified all of the possible customers of our work. Very occasionally, people would consider the differences between "novice users" and "power users." Yet some of us knew that users are as diverse as birds. *Critical thinkers refine broader categories into narrower subcategories, in order to be able to recognize important similarities and differences.* It troubled me that we lumped everyone together in this way. *Critical thinkers investigate the things that others seem to take for granted.* I tried to think of customers instead of users, and I tried to think of as many kinds of them as I could.

GETTY IMAGES

A few years ago, I read *The Inmates are Running the Asylum*, a wonderful book by Alan Cooper. The thesis of the book was that in the software industry the business people have consistently abdicated their responsibility for managing interaction design—essentially, usability. One design idea from the book had enormous resonance for me as a tester: the idea of *personas*. Mr. Cooper described his process of creating an authentic cast of characters to help guide in the design of software. Far from categorizing people as "novices," "average users," or "power users," Mr. Cooper and his team wrote entire biographies of his model users. He identified where (or if) they had gone to college, what kind of job they had, and he even went to a stock photo shop and pulled pictures to represent the characters. According to Mr. Cooper, the persona model proved to be very powerful in creating highly usable designs. Instead of designing an in-flight entertainment system for a "frequent flier," the system might be designed for Ed, a forty-six-year-old wholesale plumbing parts salesman, who flew eight times a month across his company's entire northern region; and for Lucy, a seventy-five-year-old retired hairdresser, who had only flown three times in the last six years to visit her nephew and grand-niece; and for Sally, a veterinary assistant who took an annual vacation in Trinidad with two other women who helped run the Girl Scout troop. The point of the exercise was to humanize the process: "Ed could probably handle the icons that look like VCR buttons, but Lucy might have a hard time with them."

Whether it actually worked for him in design *(critical thinkers like to see plenty of evidence and context to support claims of effective process)*, a similar technique has often worked for me. For each testing task, I try to imagine and personalize several different customers for the application. *Critical thinkers adopt ideas and techniques across disciplines.* As an example, several years ago, I was involved with the testing of an overhauled application designed for use by bank tellers. The system struck me

as confusing. On-screen instructions were written in very inflated, jargon-filled English. Things that I felt should have taken a few keystrokes took lots of deselecting of defaults, pointing and clicking, and backtracking. As I was testing, I invented for myself a number of teller personas, each of whom had a different history with the bank. One teller had been around since before there were computer terminals, another was brand new to the job but had plenty of experience with Windows programs, and another had just been hired from a competing bank, but despite having been in this country for three years, she still had rough English skills.

This approach helped me to find a bunch of design and usability bugs, but the managers dismissed all of them as unimportant. They claimed that the rough spots in the application could be addressed in training. I considered that they might have been right. *Critical thinkers try to question their own assumptions at least as quickly as those of anyone else.* But even after several weeks, I still found myself being hindered by the layout of the screens and the buttons. I still believed they were clumsy. *Critical thinkers seek evidence to confirm or refute conjectures.* I wondered what other observations I could make.

One day I found myself at my own bank branch. Several people were behind the counter—trainees and experienced staff, tellers and managers, people from different cultures, and so on. I don't know how long the senior tellers had taken to learn the system, but they sure used it quickly. In fact, they were in too much of a hurry to use the mouse; they flew through the application using the keyboard. Wait a minute—why were the tellers going so quickly? *Critical thinkers try to remain open to moments of insight, even though we don't always have control over them.* I realized that the test plan was ignoring the bank's customers—the key stakeholders for the application.

Bank customers are typically in a hurry and don't want to be unnecessarily delayed as the teller struggles with the application. One useful test would have been to time a real transaction on the old system and on the new system, not from the point of view of system response (which *was* being tested), but in terms of user interaction—how long it takes the user to perform a specific task. That kind of testing wasn't in the plan. I sneaked around a bit and did a couple of tests anyway. The older system, character and keyboard based, though not as pretty, allowed me to work through a task much more quickly. The new system, with its clunky layout, inappropriate default choices, and mouse-centric interface, took almost twice as long.

Even though the application had been designed for tellers to use, it was a part of the bank's larger mission of serving its customers. *Critical thinkers try to observe ways in which systems are parts of larger systems.* So I changed my test strategy to consider a new class of *implicit* users of the system: the customers standing in line, impatiently waiting for the teller. Although I tried to be careful, I also tried to perform each transaction as quickly as possible, as though I were under pressure. Transactions took even longer since I made silly little mistakes. Because those mistakes forced me down different paths, I made several interesting discoveries: Depending upon the order in which I performed a transaction, the customer might be charged a service charge (or not); the application might display the actual funds available after the transaction (rather than the funds available before it); and the transaction might be routed through different internal accounts, some of which would earn more interest for the bank than others would. Now that money was involved, the bank was suddenly much more interested in those design bugs I had reported earlier.

As testers, we often hear "No user would ever do that!" James Bach has a wonderful reframe: "No user **that I can think of**, and **that I like**, would do that **on purpose**." *Critical thinkers reject absolute statements and reframe them in ways that expose weaknesses in them.* When you're testing an application, ask: What might a user do by accident? What kind of user might you *not* like—a particularly dense trainee, or a hacker? What kinds of users *haven't* you thought of? How is the user's circumstance different from your own? When you're modeling a user, whose input might be valuable? Whose biases might be leading you to miss important problems?

In writing this article, it occurred to me that the bank might have had a subtle agenda that no one dared to utter: slowing down the tellers might drive the customer toward inexpensive, unsalaried, automated teller machines—in which case, the omissions in the test strategy would have been appropriate. *Critical thinkers are willing to consider unpleasant explanations along with all of the others.* **{end}**

I realized that the test plan was ignoring the bank's customers—the key stakeholders for the application.

Michael Bolton lives in Toronto and teaches James Bach's Rapid Software Testing course all over the world. Contact Michael at mb@developsense.com.