June 2007 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

TIME MANAGEMENT
What to say when the boss asks, "how long?"

DEFECT DILEMMA
Get some perspectives on the bug-tracking debate

hurry UP & wait

When Industry Standards Don't Apply

An Arsenal of Answers

by Michael Bolton

ISTOCKPHOTO

As a testing consultant, I'm often asked, "My manager wants to know how long it will take to test this product. How do I give him an answer and still know that I have enough time to test?" Providing an answer starts with understanding what the manager's question really means.

The manager might be asking "When will the testers be able to declare the product ready to ship?" This is a dangerous question, because it presumes that testers set the quality standards for the product. As testers, our role is to provide timely, relevant, important information to the project community—especially management—about how the product can work and how it might fail, so that they can make informed decisions and take appropriate actions. The state of the product is important, but it's only one part of the release decision, which is a business decision. Project managers should have information about market conditions, business needs, contractual obligations, commitments to partners, and the like. They should have authority to make decisions about project resources, schedule, product scope, budget, training, support, and deployment readiness. As testers, we contribute information to those decisions, but we don't make them.

The manager might be asking for help in making the decision: "When should I ship the product?" Framed this way, the question recognizes the manager's right and responsibility to decide. For a tester, the short answer is "whenever you like." The longer version goes like this: "I can't make your decision, but I can give you information that helps you. I'll focus on finding important problems quickly. If you want to know something specific about the product, I'll run tests to find the answer. Any time you need me to report my status, I will. If you decide to change the ship date, I'll abide by that. You can release whenever you decide that you don't have any more important questions about the product, and that you're happy with the answers you've received."

The question might be "How much time would you like to have to test the product?" This question is similar to "How much money would you like to be paid?" The answer, for most people, probably would be "as much as you're willing to give me." Testing is implicitly an open-ended search, so testing could theoretically go on forever. "Forever" is not usually a palatable answer, nor is it reasonable. At some point, the added value provided by more testing isn't justified by its cost. Moreover, decisions about quality are inherently subjective, political decisions based on whomever's values matter. Testers don't run the project or set the schedule; we respond to development work and to demand for a deeper understanding of the system. What happens will depend largely on what already has happened.

The question might be "What can we do to speed up testing?" It's wonderful when a project sponsor asks this question when the answer is going to prompt positive action. From the tester's perspective, testability is the key. Testability includes controllability—typically in the form of a scriptable application program interface to control the program via automation—and visibility—log files, on-screen monitors, or anything that lets us see what's going on. Aspects of testability also include access to information about the product and the business domain, access to the developers and other members of the project team, early and frequent availability of code, modularity of the code, and ease of setup and configuration. And there's one other crucial element: the condition of the program when we receive it.

Assume that a test for some feature takes two minutes. (This is artificial and silly—tests aren't equivalent in size, scope, and value, but bear with me.) And let's say that investigating and reporting a bug takes ten minutes (again, silly, and probably a severe underestimate). And let's assume we have sixty minutes for testing. We're given Module A to test. It has no bugs at all; in an hour of testing, we'll be able to perform thirty feature tests. Module B has a bug; we'll spend ten minutes investigating and reporting that problem. In the remaining fifty minutes, we'll be able to run twenty-five more feature tests, for a total of twenty-six tests in the hour. We find five problems in Module C. Investigation and reporting takes fifty minutes; ten minutes remain for five feature tests of two minutes each.

From one perspective, things look good; we've found five problems in Module C. In Module A we've covered thirty test ideas, and in Module B twenty-six test ideas, but the problems in Module C leave us time to exercise only ten test ideas—one-third of the feature tests that we anticipated. Either we'll have to allocate more time to this module, or leave some ideas untested. When we find numerous bugs in a product, testing slows down, coverage worsens, or both.

The question might be "When will we know that we've found the most important problems?" Since bugs are hidden, this question can't be answered; if we knew where they were, testing would be fast and easy. Some people use metrics to try to estimate the number of problems in a product. This approach is subject to a critical-thinking error called the probabilistic fallacy: The number of bugs in this product isn't predictable. Floridians aren't safe from additional hurricanes just because they've experienced the predicted number of storms for the season. Police don't conclude investigations simply because they have rounded up the average number of suspects.

Although we can't know that we've found the most important problems, we can use a pragmatic approach to find them: Provide the best test coverage we can in the time we have available. Test coverage is the extent to which we have modeled and tested the product. The more models we use, the less likely we are to miss a class of problems. Choose a diversified, risk-based, product-focused strategy to evaluate the product. Rapid testers use the Heuristic Test Strategy Model (see the StickyNotes for more information) suggests product elements (structure, function, platform, data, operations, and time) and quality criteria (capability, reliability, usability, testability, and several other -ilities). Diversify your test techniques to identify risk areas that you might not have considered.

The question might be "How long should the test phase last?" Reframe that: Is there ever a phase solely devoted to testing? When a project enters a "test phase," the testers have been testing since the project started and continue testing until release. The developers keep developing during the "test phase"; they're adding code or fixing problems until the product ships. All cycles of development are followed by cycles of testing; the "test" phase is really the fix phase.

After reframing, the project manager might ask, "OK, how long should a test cycle last?" That's a project-specific question, too, but we can apply some heuristics. The cycle should be long enough to respond to the development work that has just finished and short enough that it doesn't lag behind the next cycle of development. Time spent on reinstallation and configuration takes time away from test coverage, so cycles should be long enough to configure the product and get some useful work done. Every change to the product could change what we know about its quality; so, as the release date nears, make fewer and smaller changes per build and shorten development and test cycles.

I used to worry about not having enough time to test. Those worries disappeared when I recognized that as testers, we serve the project, rather than drive it. We keep providing service until the client is satisfied. The question isn't whether the tester has enough time—the question is whether the client has enough information, and the client decides that. When we understand management's authority, we always have enough time to test. **{end}**

Michael Bolton lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries. He is a regular contributor to Better Software *magazine. Contact Michael at mb@developsense.com.*

Sticky Notes

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

- The Heuristic Test Strategy Model

What alternate meanings to the "how long" question have you encountered?
How did you respond?

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

S
S
S
S
So...
You're Looking For An Open Source Automation Solution?

That's what we thought. And that's why, at ACULIS, we created, APODORA, a functional test automation tool. Now we would like to share it with you! Curious? Well you have a few choices: you can simply turn the page for more info, or visit www.apodora.org to download the latest version, or you can contact us directly at:

SOFTWARE DEVELOPMENT SERVICE SOLUTIONS | DEVELOP IT TEST IT GLOBALIZE IT STAFF IT

ACULIS

HQ: 801.377.5360 TF: 866.4ACULIS
URL: WWW.ACULIS.COM