The Testing Team's Motto

Brian Marick, marick@rstcorp.com
Return to Marick's Corner

We are a service organization whose job is to reduce damaging uncertainty about the perceived state of the product.

That's my motto for a testing team. Although it doesn't have quite the zing of "Be Prepared", I chose the words carefully to remind testers of aspects too easy to forget. I explain it below.

At some point in a project, someone has to make the decision whether the product is ready to ship. I'll call that person the "project manager", though it might be the VP of Engineering, the CEO, or someone with some different title. It's a hard job. It requires answering many questions:

- "If we slip, will we miss the PC Magazine annual Widget issue? And how much damage will that do to our sales?"
- "Are the programmers so burnt out that further work poses a substantial risk of introducing more problems than it solves?"
- "Exactly how buggy is the product, anyway?"

The answers to those questions are always uncertain. The testing team's job is to *reduce* the project's manager's *uncertainty* about the answer to the last question. But there are different ways to reduce uncertainty, some more useful than others:

- The testing team could concentrate very hard on one half of the features of the product. Uncertainty about those features is at a bare minimum. Unfortunately, no one knows anything about the other features.
- Or they could spread the same effort more evenly. They will be less certain about half of the features, but more certain about the other half.

Put that way, the choice seems easy. The risk of shipping a completely untested feature is far too high. But more subtle types of misdirected effort occur all the time in testing, unless test plans are devised with the goal of reducing the project manager's *damaging* uncertainty about the product, the uncertainty that makes good decisions harder.

What about finding bugs? Isn't that the job of testing? Yes. The way you reduce uncertainty is by a well-directed and relentless search for bugs. The right bug reports will lead to the right decisions.

But notice that my motto doesn't refer to discovering the bugginess of the product, but rather its *state*. "Bugginess" has too narrow a connotation. It usually refers to a product that doesn't do what was intended. But what if the product works exactly as intended, but what was intended is wrong? It doesn't meet real customer needs, or it's too hard to use. Testers have a unique and useful perspective on those issues - that of the expert user. My point in using the word "state" is to make sure that all important problems get in the bug database.

Ah, but what's "important"? It is what the eventual customers *perceive* it to be. That can be a facile statement ("The customer is always right. Good, now we can stop thinking about them."), or it can be a guide to action. The good tester tries to understand the target market and the typical customer. She talks with customer service. She strives to report bugs persuasively, in a way that makes clear how many customers will be affected and to what degree. Again, this awareness helps direct testing efforts toward the bugs that matter. For example, do you know how important small bugs are to your customers? In some markets, customers shrug off bugs with simple workarounds; in others, such a product may be deemed too buggy - even in comparison to one that fails less often but more spectacularly.

Let me make a final note about testing as a *service organization*. I've claimed it serves the project manager. By facilitating good decisions, that helps the customer, but only indirectly. Most everyone on the project serves the customer more directly than the testers do. The programmers produce the code that the customers use. The project manager decides what code that will be - and which of it will remain buggy and which will be fixed. The writers produce the words that customers read. The customer service people directly help customers with problems. Almost all of what testers produce is invisible to the customer (except for known bug lists in README files and the like).

Many testers find this claim a disturbing way of thinking. They think of themselves as the only protection the customer has against careless developers and managers determined to meet deadlines no matter what. They desire the authority to stop shipment of the product if it doesn't meet certain predefined quality criteria. There are problems with that idea:

- Do testers have enough knowledge? Can they answer the questions in the first bullet list, along with all the other questions a project manager should ask? Usually not.

- "Stop ship" authority is usually fictional. When testers try to exercise it, they are often overruled. They become demoralized and ineffective.

- By taking on the responsibility for minimal quality, they relieve others of it. The result may be a worse product, as quality drifts down toward that minimum.

By not taking on an impossible responsibility, testers will become more effective at the job they can do - that of serving the project manager. Oh, they still have stop ship authority in exactly the same sense that all project members do. They can try to persuade the project manager. If the situation is serious enough, they can vote with their feet by resigning. If the situation is very serious, they have the same ethical responsibility to blow the whistle as anyone else does.

I don't mention the project manager in the motto because the testing team also serves the rest of the project, though to a lesser degree. For example, programmers often have a poor sense of the state of their code. Even though they get information in the form of bug reports, they are too close to their code to put it in perspective. They may believe that serious bug reports are just annoyances from the tester, simply because they don't have a good picture of what customers really do. Or they may spend their time honing and polishing what isn't important at the expense of what is.

When I work with developers, I tell them in the beginning that my job is to tell them what they need to know to get their code working as soon as possible. They should tell me if they think I'm reporting the wrong bugs, too trivial bugs, or could in any way be helping them better. I reserve the right to disagree, but I certainly want to know what they think. My experience is that most developers are happy to hear that. Oh, some fraction of them take it as a signal they can roll right over me, but I think not many more than would try that no matter how I approached them. Relations with the others start off better, which quickly results in more efficient bug reporting and fixing. At some point, most programmers become evidently overwhelmed by the flow of bugs and wish - under their breath - that I would just leave them alone. I maintain an attitude of cheerful relentlessness: "I'm going to help you no matter how much it hurts".

Which, come to think of it, would make another good motto for the testing team.