Deception and Self-deception in Testing

Fiona Charles

Originally published on stickyminds.com May 28, 2009.

Have you heard any of these lately?

> "The testers are finding too many bugs and holding up the project."
>
> "Anyone can test. We just have to give them the right process to follow."
>
> "Our test cases will provide complete system coverage."

Not one of these common statements about testing is true. At least one of them could have been said by a tester.

Delivering and promoting accurate communications about testing is essential to the tester's and test manager's job. We have a responsibility to dispel myths and misconceptions about good testing and what it can and cannot do. We must also be alert to and prepared to address distortions or attempts to spin the message about testing from any source—including ourselves.

Testing deceptions and self-deceptions often arise from excessive optimism—the triumph of hope over experience (or hope over hard data). Sometimes they come from people attempting to find a place to lay blame. Humans can fool themselves into believing all sorts of impossible things, and occasionally they even resort to deliberate lies. Exaggerating or downplaying risk, inflating test coverage, blaming testing for project delays when the product quality is poor, misrepresenting testing status and findings—these are only some of the kinds of deceptions and self-deceptions testers encounter on software projects.

Let's look at some more typical examples.

Deceptions Practiced on Testers

> "The software is done. It's ready to test."

Every tester has heard this one, only to discover that the meaning of "done" and "test-ready" doesn't reflect what was in the project plan to get done by this date. Somehow, little things like unit tests—sometimes even finishing coding on some modules—have ceased to be requirements. The programmers made the date, so they're "done."

We've all heard plenty of others. Here are some common ones:

> "We didn't change anything significant. You don't need to test." *[Although nobody did an impact analysis, and we don't actually know what might break.]*

> "The infrastructure upgrade *[that includes the operating system, the database engine, and the compiler version]* will be transparent to the applications. You'll only need to do a sanity test."
>
> "You only need three weeks for testing." *[Because the code is late and we cut three weeks from the test schedule.]*

Are programmers, project managers, and others lying when they say these things? Quite possibly not, but if not, they've certainly fooled themselves into believing what they want to believe.

Deceptions about Testers and Testing

Sometimes deceptions appear at a higher level, in what managers, project managers, and programmers say about testers and testing. Often, those untruths reflect what they actually believe:

> "The testers don't know what they're doing." *[They estimated it would take two weeks, but found incomplete code and so many bugs it has taken six, and they're not done yet.]*
>
> "Our mature test process employs all the industry best practices." *[Major bugs frequently bring production systems down, but we have all the "right" testing documentation.]*
>
> "Total test automation will make testing much faster and more efficient, and we can save on expensive labor costs." *[We haven't thought about who will design the automated tests, and we have no idea what it will take to maintain them.]*

Deceptions Testers are Pressured to Practice

Testing can be the first place where cracks in a project appear. The light we shine on product quality isn't always welcome, especially when it illuminates a midden full of problems.

Managers who have been hiding ongoing quality issues, or deceiving themselves into not seeing them, can be thrown into a panic by test results that show poor product quality. Some will be tempted to skew the results they report upwards, putting strong pressure on testers to misreport their findings—to show testing progress or product quality as better than it is:

> "We have to get creative about these numbers." *[And paint a false picture.]*
>
> "We can call more of these tests passed." *[Although they have failed, or perhaps not even been run.]*
>
> "The rest of the project is status green. Testing needs to be green, too." *[Though it clearly isn't.]*

These are some of the things testers will hear from managers who want them to lie.

Deceptions Practiced by Testers

Testers are no more immune than anyone else to temptations to fool ourselves and others about testing. We can easily succumb to the belief that we are doing the right thing—or the only possible thing—when we have blinkered ourselves to other possibilities. And we too like to believe that:

> "We're going to make it!" *[Whatever "it" is—often an impossible date with impossible scope.]*

Have you ever exaggerated the testing risks to get the time you thought you'd need for testing? Or padded your estimates because you "knew" you'd need contingency you couldn't get into the schedule otherwise? How about overplaying test coverage or inflating the efficacy of your team's ability to find bugs with your testing process?

Or maybe you downplayed a material fact. At a workshop I conducted recently on testing deceptions, testers said they frequently ignored bugs that were "out of [their] scope," saving time in reporting by convincing themselves that these bugs weren't important enough to bite somebody sometime.

Testers' deceptions are no less damaging—to the software and themselves—than any other kinds of deception.

What Can We Do about Testing Deceptions?

In every case, the answer is the same: Ascertain the facts, and tell the truth. That's simple to say, but of course it isn't always easy to do. Combating the self-deceptions of others can be particularly difficult. And people, especially managers, can have a lot invested in what they've said. But we have to try.

Here are some steps to help avoid deceptions:

- **Find out the facts, preferably before a deception occurs.** Those facts might be about the software quality, the pros and cons of test automation and the various options available, or why the product quality is holding up the testing—and therefore holding up the project.
- **Present the facts straightforwardly and unemotionally**.
- **Never give in to pressure to lie.**
- **Be scrupulous in our own thinking and communications about testing.** That means being open about our thinking and planning, and reporting the risks and problems we find.

Testers are paid to provide information about software quality. It's our job to tell the truth about testing and to see that misconceptions aren't proliferated.

What's your experience with deceptions or self-deceptions in testing? Have you ever fooled yourself—or maybe even consciously deceived? Have you had to address

deceptions proffered by others? How would you advise your fellow-testers to deal with deceptions?