The Sceptical Tester

Fiona Charles

Originally published on AGILE CONNECTION, May 21, 2012.

The Sceptical Tester
By Fiona Charles

Summary: Testers are people who ask questions, think critically about the answers, and then ask more questions—repeatedly. Fiona Charles reminds testers that their success depends on maintaining a healthy scepticism.

Someone asked me recently what is the worst mistake a tester can make when starting out on a project. I didn't have to dig very hard for my reply: "Believing what you're told, and accepting what you're given." OK, that's two mistakes. They're inseparable.

Do I mean that every tester should go around vibrating with suspicion and saying, "Don't give me that!"? No, of course not. That would be neither professional nor practical.

I mean that you should never assume that what you have been told is the whole truth and nothing but the truth. Never believe that the documents you've been given contain all the information you will need to absorb and understand for testing the system. Never take as fact anything you haven't taken trouble to question and, if possible, verify. Where you can't verify and you need to move on, make working assumptions and then revisit them periodically to see if you've since learned anything that would prove or disprove them.

Testers are people who ask questions. We are sceptical people who examine critically all the information that comes our way and then go looking for what's missing. We know critical questioning is essential to our work because the information we need is often scattered, buried, or not really known by anyone on the project. And sometimes projects nurture myths—commonly held beliefs that can obstruct, yet may collapse when probed.

On one big program where I was program test manager, everyone knew that all test personnel must have a particular certification. It was in the contract with our government client. My manager told me this when I started (although I possess no certification and they engaged me), and all my leads believed it. I was concerned that this requirement would get in my way, but I had bigger problems to deal with and at least I seemed to be getting good functional testers.

But then I needed performance testers. Week after week I met with the hiring manager only to learn that we still hadn't got the skilled people my performance test lead urgently required. I discovered that our recruiter was routinely turning away highly qualified performance testers on the sole ground that they weren't certified.

This was crazy. Even in markets where most testers have certifications, performance testers typically don't bother to certify. If we really were contractually bound, we would

need to seek an exemption from the client. So I asked one of my leads to search our large and complex contract for the certification clause.

When finally unearthed, the clause read, “certification or equivalent experience.” So much for received belief! When we stopped acting on myth, we had no trouble recruiting the skilled people we needed.

On another project, I almost hit a showstopper through believing what I was told. To test the end-to-end integration of a new e-commerce site, I needed to have a coordinated dataset across a big suite of applications. None of the many planning, supply chain, or merchandising systems upstream from the online store was being changed for the project. We just needed to get product, pricing, and promotions data from them in the end-to-end test. The teams responsible for those systems were used to testing together, and they assured me they had a coordinated product set for use in all their integrated testing. They’d be ready to participate in the end-to-end test when I needed them.

Great! I focused my efforts on other urgent tasks. The online order management system and warehouse management system were new, and there were myriad associated changes to the many systems downstream, including backend financials.

But some time later, I learned that the product set with which the e-commerce site was going to launch contained mostly items that the upstream systems lacked in their test data. Preparing a new coordinated dataset for the upstream systems would be both time-consuming and expensive. The managers of those teams balked at taking on the work, yet the end-to-end test was considered critical for successful launch of the strategic e-commerce site.

Ultimately, I found a way to segregate the upstream systems and their existing data without compromising the critical flows in the end-to-end test, but it was a close call. I hadn’t asked enough questions early enough about the data, and it could have killed my test.

At two other clients, I was told that the general ledger (GL) system was out of scope for the project, and hence for the end-to-end test. It’s understandable that managers would stipulate this. The more systems you test with, the greater the cost. Why include the backend financial systems if nothing there has changed? I didn’t accept this in either case, and I gently probed until I confirmed my suspicions.

On the first project, where we were developing a new bank teller system, the solution architect had left the GL off the system context diagram because it was “not needed.” But our teller system had a direct batch interface to the GL, and it didn’t make sense to me to leave it out. In our first integrated test with the GL, we found that all the credits and debits from our system were reversed—an easy bug to fix, but it hadn’t been detected until we tested with the GL. Through continuing to incorporate the GL in our test, we found several more obscure and complex bugs in our system.

I discovered an even more interesting situation on the second project, which involved ordering, provisioning, and billing for telecom services. Since there would be new financial outcomes, I naturally included the GL system in my end-to-end test scope. But when I reviewed my strategy with the project sponsor, he said it was out of scope—no work being done—and I should remove it. That didn’t feel right to me, and I said I’d like

to do a little more investigation before definitely ruling it out. “Well, OK,” he said, “but don’t waste too much time on this. It’s not in scope.”

There was a solution review coming up, so I shelved the question till then. It was a three-day whiteboard session, with each team drawing its systems and data flows, and the integration architects and me asking questions. When the team responsible for the billing and financials had done its diagram, I said, “I must be missing something. I don’t see a line out to the GL, but I thought we were dividing the accounting between two lines of business.”

“That’s right,” their architect replied, “and we’ve created two new sets of GL accounts.”

“Great,” I responded, “but where are we actually dividing the revenues and doing the accounting?”

There was a silence, and then their project manager spoke up. “Um, I think we might have forgotten that. I’ll look into it and get back to this group.” The result was a $250,000 change request. The GL was definitely in scope.

Testers ask questions and then question the answers. It’s our job.

Think about this the next time you start a new project. Or stop for a moment and think about the questions you haven’t asked on your current project. What have you been told that might be wrong or not the whole story? Are the requirements documents you’ve been given the most reliable source of information about the system? Where else should you look? Who else should you talk to?

I took the examples for this article from big projects, but the lessons apply to projects of any size, whether you’re managing an enterprise end-to-end test or you’re the sole tester on your project team.

Never believe everything you’ve been told. Don’t accept that what you’ve been given is everything you need. Your success—and that of your project—depends on your healthy professional scepticism.