Not Wanted on the Voyage

Fiona Charles

Originally published on StickyMinds.com February 19, 2009.

Before intercontinental air travel became ubiquitous, people crossed oceans by ship. The things they needed for practical, everyday use traveled with them in their sleeping and living quarters. Other baggage—typically consisting of heavy and cumbersome items like steamer trunks—was labeled "Not Wanted on the Voyage" and consigned to oblivion in the ship's hold until the end of the journey.

In an earlier column ("What's In a Name?") I wrote about my concern that testers are often second-class citizens in the software world. I had just been hearing yet again that many agile teams don't value testers—that to them testers are heavy baggage and Not Wanted on the Voyage. But agile teams aren't the only people to have issues with specialist testers and their practices. Many managers, too, believe that testing is excessively slow, expensive, and not nearly effective enough for the realities of their delivery projects.

Some reasons for negative perceptions about testers and testing are beyond testers' control. And some of the conclusions are just plain wrong, or misguided at best. But in this column, I'd like to explore our own responsibility. What are the things that testers do or don't do that lead some software practitioners and managers to believe their projects are better off without us? Do we have beliefs and practices of our own that may be getting in our way?

Process vs. Skill

Too many of us put too much reliance on process and not enough on skill. In many organizations, before any testing can begin, testers must develop detailed test cases or scripts from frozen, documented requirements and then get them signed off. In some places, it's routine for "testers" who have no actual testing skills—or even business domain knowledge that might compensate a little for the lack—to execute test cases written by others.

I have managed many test efforts where it was appropriate—even essential—to the context to use predesigned test cases. Although, I have always insisted that testers execute tests they have researched and designed themselves. But this process is time-consuming, expensive, and flawed. It is not the most effective way to find bugs as other writers have said, and my teams have demonstrated many times with supplementary exploratory testing. Perhaps worst of all, it assumes a quality level in documented requirements that I don't remember ever seeing in thirty years.

When this is the only kind of testing done in an organization, is it surprising if managers don't see benefits commensurate with the costs?

We need to be more flexible. The "traditional," structured test process is neither the only right way to test nor the only disciplined way. It should be one option among many, used

only when it's clearly the right thing to do in the circumstances. But that means that all testers need to educate themselves and their organizations about other valid methods.

Operating with flexibility rather than routinely following a structured process demands more of testers. The flexible tester needs skills to evaluate project contexts for the appropriate methods and techniques to use. That's only possible if the tester has a broad knowledge of what is available and the skills to execute the chosen methods and techniques.

Collaboration and Teamwork

Agile projects depend on constant collaboration among team members. There's nothing new in that; it's true of all successful software projects.

Yet often, testers aren't encouraged to work collaboratively with programmers. Some don't see themselves as part of the same team. At StarWest 2008, there was even a keynote with the title "Has the Time for the Adversarial [Test] Organization Passed?". Seeing this on the program suggested to me that it isn't unusual for software professionals to think of test organizations in this way.

Was there ever a time when it made sense for testers to be adversarial?

Independence is important for testers, and there can be good reasons for a test group to have a separate reporting structure from that of the programming organization. Organizational independence can help to ensure that testers' needs for time and resources are heard and that their findings aren't buried. I believe it played a part, especially in the early days, in testers' ability to develop a professional identity and discipline.

But independence doesn't require "us and them" relationships. Do we think programmers should want to work with adversarial testers? Would you if you were a programmer?

True independence is independence of mind. Testers need the courage to test what needs to be tested and to give accurate information about our findings, but we also have to work with programmers, not against them. Our jobs are different, but we share with programmers both the activity and the goal of developing working software.

Gatekeeper Mentality

In many companies, the testers "certify" that software is ready for release. Sometimes the responsibility has been forced onto testers by management, but more often the test organization wanted it and even fought for it. Why?

I've found in my consulting that testers in gatekeeper roles typically feel beleaguered, often acting as if they are the last bastion of quality before less-than-perfect software hits an unsuspecting world. Being gatekeepers can cause testers to become self-righteous and judgmental, behaving like the quality police (or judge and jury), blaming programmers for software bugs, and acting as if no one else cares about software quality. Some testers feel crushed by the responsibility. They can become obsessive, afraid to look objectively at their testing, and reluctant to let go of the software even when they aren't finding bugs.

In reality, testers know we can never really test thoroughly enough to certify anything.

The best we can do is to describe the testing we have and have not done and what we have and have not found out about the software. We can apply our best professional judgment to describing the significance of our work and our findings, and we can categorize that significance in terms of risk—while understanding that we are not always in the best position to estimate the business impacts of software problems.

The tester's job isn't to judge. It's to provide information about the software to help management make informed decisions. That's a difficult, honorable, and challenging job. We shouldn't denigrate it by becoming gatekeepers.

Let's Get Rid of Them!

I don't think testers want to be labeled Not Wanted on the Voyage. We want to contribute real value to software projects with our unique testers' mindset and skills. To ensure that future, we all need to keep growing what we do and how we do it.

A culture that values process over skill, an adversarial attitude, and a gatekeeper mentality are heavy baggage for testers to carry around. They hold us back from doing our best work. They alienate our teammates and perpetuate beliefs that we are inflexible and incapable of adaptation—a poor bet for shipmates on challenging voyages.

I'd like to slap Not Wanted on the Voyage labels on all these counter-productive practices and beliefs. But let's not store them in the hold for future use. Let's throw them overboard! Or quietly leave them behind on the quay.