AM I CREATING VALUE WITH MY TESTING?

by Jonathan Kohl | www.kohl.ca

This article was originally published in the Star Tester Magazine, November 2009.

Jerry Weinberg says: "*Quality is value to some person.*" As software testers, we can relate to that statement. The software we test must provide value to the people who use it. But how often do you turn that statement onto your own work? Even if you're following generally accepted testing practices, are you creating value for the people you serve?

I had to learn this the hard way. Early in my testing career, I was assigned to a project as a test lead. I was excited to lead test planning, test design and execution. I started a test plan document, determined existing regression tests I could use, and created test cases based on the requirements.

We had recently developed new Quality Assurance processes and were pleased with our results. We had progressed from a chaotic environment to a well-defined, repeatable testing process and were especially proud of the large number of procedural test cases we had written. Our tracking software made it easy for us to get metrics on what was covered. Our favorite was "percent complete." In status meetings, we could reply: "we're 75% complete" with a smile on our faces. We were confident that we had covered 75% of the test cases we planned to run. We could also metrics on bugs and their severity.

Most of stakeholders on the project were relieved when they heard the metrics. It seemed to make them feel better about the testing progress. I felt better too. I thought I was making headway on the project.

The development manager, however, was disappointed in my work. As I announced, "I'm 75% complete," he replied with, "How can you know you're 75% complete?" So, I showed him our test cases and how we tracked them. "How can you know you're *any* percent complete?" he asked. I mumbled something about test cases. He said, "You can only tell me what percentage of test cases you've run of the ones you have *thought* of. There are more you haven't thought of, so that number is practically meaningless." I was proud of the test cases I had written. They were very detailed with a concise test case title, a purpose so we would know what we were testing, the steps to execute, and the expected results. He was unimpressed.

"You're only tracking the tests you have developed, and I don't have much faith in them at all! You testers are all caught up in this process and aren't *testing what's important*. You've written tests off of the requirements, but that only verifies what we've already done in development." He went on to say the quality of our work was suffering due to our procedural test case obsession. Wow! That stung. But, I realized he was right.

At that time, I hadn't worked in testing for very long, but I had a lot of experience in part-time customer service jobs. I had a job at a popular local restaurant while I was in university. There management had an obsessive vision to satisfy and impress customers. They hired and trained skilled staff who were instilled with this vision. Every role in the restaurant, from the hosts and hostesses at the door to the servers, cooks and dishwashers, was focused on impressing the customer.

Management reminded us daily how our roles fit into the big picture, and how the specific tasks that we undertook mapped to great customer experience. We were also given latitude in our work – if we came up with a better way to do our jobs, they wanted to hear about it and share it with the rest of the team. The team created a clean, stylish, pleasant atmosphere with spotless table settings polished silverware, and food made with carefully with the freshest ingredients. Service was prompt, friendly, and the servers were highly skilled, which impressed even the most difficult customers. The end result was a restaurant that was busy, provided good value, and had many repeat customers and regulars.

After that run-in with the development manager, I contrasted the restaurant with work on software projects. I was surprised at how little we knew about the customer when we were building software systems that were far more expensive than a meal. It was difficult to find anyone on the technical team who could tell me much about our customers, let alone tell me how any work we did contributed to satisfying their needs. Instead, we focused a lot more on software development methodologies, processes, technologies and tools. Sometimes, we even deferred our responsibilities onto the customer: "They didn't ask for security, so we're not going to build that in!" In the restaurant, we weren't asked to have food that met regulations and was safe to eat. Rather, the customer depended on us to use our technical knowledge, experience and skill to know how to do those kinds of things properly so they wouldn't need to worry.

Drawing from my restaurant experience, I overhauled my software testing approach for the project. I started asking stakeholders in the company how they felt our products delivered value to our customers, and what they expected from our testing efforts. I talked to customer support, development, marketing and sales, and even senior management. With that insight, I reprioritized my testing and created a test strategy with a clear goal.

I decided to focus more on test execution and to limit the clerical overhead of test cases and management systems. I used more exploratory testing to expand our test coverage and looked to the work of Cem Kaner and James Bach on how to manage and report that effort. I converted many regression tests into checklists, preserving traceability with more flexibility. Moving beyond functional and requirements-based testing, I used more models of coverage. I determined these through technical and business-related investigation of the product. Some testing was much more technical, and some mapped directly to the way our end users interacted with our software. We weren't just repeating what the programmers were testing anymore.

To help speed up tasks and be more efficient, I worked with another tester to create task automation scripts. These would automatically install builds, monitor logs, and set up testing.

I changed the way I reported testing status to reflect both qualitative and quantitative elements. I included reports on different models of coverage. If I used a count or percentage, I qualified it with an explanation of what the numbers meant, and what their limitations were.

Most importantly, I regularly asked stakeholders for feedback about my testing service and adapted it where needed. I knew I was creating value with my testing. I was finding important problems and testing much more thoroughly than before. At the end of the project, the now smiling development manager told me it was the most thoroughly tested, well-reported project to date.

Ever since that experience, I try to figure out how to best create value with my testing. My approach can change from project to project, and from organization to organization, but the question is always the same: "Am I creating value with my testing?" And, if not: "What do I need to change?"

About the Author

Based in Calgary, Alberta, Canada, Jonathan Kohl is the founder and principal software consultant of Kohl Concepts, Inc. A noted software testing thinker and strategist, Jonathan is a natural investigator on software projects. In addition to assisting teams with testing, Jonathan helps companies define and implement their product vision, coaches practitioners as they develop software on teams, and works with leaders helping them define and implement their strategic vision. Jonathan is also a popular author and speaker. His blog on software development and testing issues is one of the most well-read testing blogs in the industry. Jonathan is a regular contributor to Better Software magazine, both as an author and technical editor. Contact Jonathan at http://www.kohl.ca/.