RBCS
TIME TESTED. TESTING IMPROVED.

Measuring Confidence along the Dimensions of Test Coverage

By Rex Black

When I talk to senior project and product stakeholders outside of test teams, confidence in the system—especially, confidence that it will have a sufficient level of quality—is one benefit they want from a test team involved in system and system integration testing. Another key benefit such stakeholders commonly mention is providing timely, credible information about quality, including our level of confidence in system quality.

Reporting their level of confidence in system quality often proves difficult to many testers. Some testers resort to reporting confidence in terms of their gut feel. Next to major functional areas, they draw smiley faces and frowny faces on a whiteboard, and say things like, "I've got a bad feeling about function XYZ." When management decides to release the product anyway, the hapless testers either suffer the Curse of Cassandra if function XYZ fails in production, or watch their credibility evaporate if there are no problems with function XYZ in production.

If you've been through those unpleasant experiences a few times, you're probably looking for a better option. In the next 500 words, you'll find that better option. That option is using multi-dimensional coverage metrics as a way to establish and measure confidence. While not every coverage dimension applies to all systems, you should consider the following:

- **Risk coverage:** One or more tests (depending on the level of risk) for each quality risk item identified during quality risk analysis. You can only have confidence that the residual level of quality risk is acceptable if you test the risks. The percentage of risks with passing tests measures the residual level of risk.
- **Requirements coverage:** One or more tests for each requirements specification element. You can only have confidence that the system will "conform to requirements as specified" (to use Crosby's definition of quality) if you test the requirements. The percentage of requirements with passing tests measures the extent to which the system conforms.
- **Design coverage:** One or more tests for each design specification element. You can only have confidence that the design is effective if you test the design. The percentage of design elements with passing tests measures design effectivity.

- **Environment coverage:** Appropriate environment-sensitive tests run in each supported environment. You can only have confidence that the system is "fit for use" (to use Juran's definition of quality) if you test the supported environments. The percentage of environments with passing tests measures environment support.
- **Use case, user profile, and/or user story coverage:** Proper test cases for each use case, user profile, and/or user story. Again, you can only have confidence that the system is "fit for use" if you test the way the user will use the system. The percentage of use cases, user profiles, and/or user stories with passing tests measures user readiness.

Notice that I talked about "passing tests" in my metrics above. If the associated tests fail, then you have confidence that you *know* of—and can meaningfully describe, in terms non-test stakeholders will understand—problems in dimensions of the system. Instead of talking about "bad feelings" or drawing frowny faces on whiteboards, you can talk specifically about how tests have revealed unmitigated risks, unmet requirements, failing designs, inoperable environments, and unfulfilled use cases.

What about code coverage? Code coverage measures the extent to which tests exercise statements, branches, and loops in the software. Where untested statements, branches, and loops exist, that should reduce our confidence that we have learned everything we need to learn about the quality of the software. Any code that is uncovered is also unmeasured from a quality perspective.

If you manage a system test or system integration test team, it's a useful exercise to measure the code coverage of your team's tests. This can identify important holes in the tests. I and many other test professionals have used code coverage this way for over 20 years. However, in terms of designing tests specifically to achieve a particular level of code coverage, I believe that responsibility resides with the programmers during unit testing. At the system test and system integration test levels, code coverage is a useful tactic for finding testing gaps, but not a useful strategy for building confidence.

The other dimensions of coverage measurement *do offer* useful strategies for building confidence in the quality of the system and the meaningfulness of the test results. As professional test engineers and test analysts, we should design and execute tests along the applicable coverage dimensions. As professional test managers, our test results reports should describe how thoroughly we've addressed each applicable coverage dimension. Test teams that do so can deliver confidence, both in terms of the credibility and meaningfulness of their test results, and, ultimately, in the quality of the system.

Bio

With a quarter-century of experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For over fifteen years, RBCS has delivered consulting, outsourcing and training services to clients ranging from Fortune 20 companies to start-ups. Rex is also the immediate past President of the International Software Testing Qualifications Board and the American Software Testing Qualifications Board. Rex has published six books which have sold over 50,000 copies, including Japanese, Chinese, Indian, Hebrew, and Russian editions. He has written over thirty articles, presented hundreds of papers, workshops, and seminars, and given about fifty speeches at conferences and events around the world. Rex may be reached at rex_black@rbcs-us.com.