Investing in Software Testing: Maximum ROI Through Pervasive Testing

Abstract

Pervasive testing means getting the right people working together through the right processes at the right time for high-ROI testing. Pervasive testing is the final piece of the test investment puzzle, and just as important as all the others. Through pervasive testing, all the ideas we've explored so far come together.

Introduction

Does your organization thrust the responsibility for testing on a small cadre of people, secluded in a dimly-lit test lab, right at the end of the project? Even with the best tools and techniques, such a team can't create the kind of return on investment I've been telling you about throughout this series of articles. Successful test efforts start early, involve all appropriate stakeholders and participants, and come about through cross-functional teamwork. Like the roots of a potted houseplant spreading through soil, successful test efforts *pervade* the project.

Pervasive Testing Timelines

When testing pervades a project, the test tasks start on the first day of the project. Test team members are involved in project planning and requirements definition. In parallel with these activities, testers organize a quality risks analysis for all key stakeholders, as discussed earlier in this series. Based on the decisions made in that analysis of what features and functions of the system require the most testing, the test manager prepares an estimate and a plan for the test effort. Once the plan and estimate are approved, testers proceed to configure the test environment and create the test cases, data, and tools at the same time that programmers are creating the system.

A graphical view of what a time-pervasive test effort looks like is shown in Figure 1. The test task bars are black, while the other project-related task bars are gray. During planning and system development, a parallel test task exists. These test tasks often focus on project documents. For example, quality risk analysis focuses on the requirements specification. At the same time as they use such documents to plan and build their tests, testers also identify potential problems, saving the project team time and money through early defect detection and removal. Early versions of the test cases and data are available to the programmers during development. Component testing, primarily structural, starts early in the project, as soon as the first components are coded, providing even more opportunities for early—and cheap—bug detection and removal. As communicating pairs of components complete component testing, integration testing begins. Once integration testing is done, we are ready to start system testing. In this way, bugs are found as early as possible, when it's cheapest to fix

them. Lowering the cost of internal failure improves the return on the testing investment even more than reducing the cost of external failure discussed in the first article.

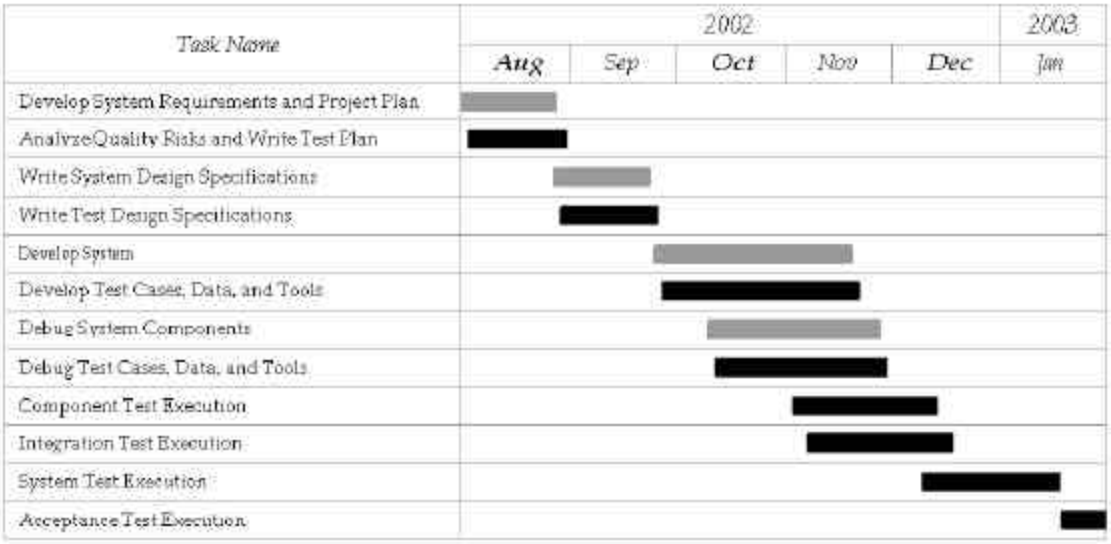

Figure 1: A Timeline for Pervasive Testing

Pervasive Testing Participants

This kind of pervasive testing can only happen when all the right people pitch in. Senior management and executives, of course, must make the commitment to start testing early. As testers armed with the knowledge that early testing is a smart investment, we can convince them of the benefits.

During quality risk analysis, all the right people must get involved. Sales and marketing people in the mass-market world, and business analysts and users in the IT world, must help define and prioritize the quality risks. Customer support or help desk staff must be involved, too.

Some test execution work will fall to developers, such as structural component and integration testing. (For a case study of what happens when developers abdicate their testing role, see Elisabeth Hendrickson's article, "More Testing Worse Quality," on her Web site, www.qualitytree.com.) Independent test teams generally focus on behavioral integration and system testing. However, re-use is possible—and important. On one project, my test team used some test drivers written by programmers for component testing to create load generators for stress and performance testing. On another project, developers used my test team's automated functional test system for regression testing as they added new functionality to the software they were working on. Pervasive testing means everyone participates in testing, and testers leverage the skills, knowledge, and outputs of the whole team to reduce costs of conformance (i.e., testing) as well as costs of nonconformance.

Pervasive testing also means that, as test execution begins, the findings of the testers play a key role in high-level project status meetings. The product of the test team is information. The information products of a pervasive test team provide the project management team with key insights into the quality of the system, and the trends playing out in system development. How quickly are we fixing bugs? What problems remain? Have we explored all the key risks yet? When will we be ready to ship? Pervasive test efforts have answers to these questions, and those answers enlighten the decision-making process of the project management team.

Pervasive Testing Teamwork

As you might expect, having all these people participate in testing means coordination—i.e., teamwork. Many of you have probably seen a youth soccer match played by children who have not yet learned teamwork. The game is a wild, disorganized stampede. Little legs and bodies chase the ball around the field, with lots of screaming and yelling. Every now and then a lucky kick sends the ball zooming off to a new part of the field, and, in a mad dash, the mass of excited children tear off after the ball, trailing loud, impatient noises.

Does this remind you of your last project? If so, teamwork is what's lacking.

With teamwork, each project team plays a specific part, and each team member knows his or her roles and responsibilities. People stretch to help their teammates, true, but each person focuses on doing their part. Sales, marketing, and business analysts define the expected uses. Customer support and help desk staff explain the key problems that exist in the current product. Programmers write code. Database administrators define schemas. Configuration management engineers maintain the code repository and create software releases. System administrators support development and test environments. Testers evaluate system quality and deliver useful, unbiased information in a timely, accurate, credible fashion. The handoffs between each team happen on time, as planned and committed. Throughout it all, managers provide resources and guidance, pro-actively course-correct to prevent crises, and promote teamwork within and across their teams.

Investing in Testing: Payoff Summary

When testing pervades project efforts, we now have the final piece of the puzzle in place to make our investment in testing pay off handsomely. The early, cross-functional teamwork with *all* stakeholders across the project maximizes our investment. To bring this series of articles to a close, let me tie pervasive testing together with some of the other topics we've covered.

Notice that pervasive testing means that the project team deploys the right resources—human and otherwise—to run the right tests at the right time using the right techniques. Pervasive testing includes early, cross-functional reviews of requirements, designs, and code—i.e., static testing. Pervasive testing includes developers running structural component and integration tests, often using

automated test tools. Pervasive testing includes an independent test team capping the mix of test techniques with behavioral testing during integration and system test. Pervasive testing is key to using the right tools and techniques at the right time.

Through the cross-functional participation of all quality and testing stakeholders across the project, developers and testers can craft structural and behavioral tests that cover the critical quality risks and test the important usage profiles. Through cross-functional quality risk analysis, we can begin to understand what it would mean for our system to have quality. In other words, what are the features, behaviors, and attributes that would satisfy—and dissatisfy—our customers and users, and how can we test for them? Without pervasive testing—early, cross-functional, and coordinated—testers struggle—and often fail—to gain the insights needed to target testing this way.

But when all the pieces come together—the right people, the right processes, the right time, the right techniques, the right focus—then we can achieve truly impressive returns on our testing investment. Significant reductions in post-release costs are ours for the taking with good testing. In cost of quality parlance, we invest in upfront costs of conformance (testing and quality assurance) to reduce the downstream costs of nonconformance (maintenance costs and other intangibles associated with field failures). For more on the financial aspects of testing—among a great many other topics—please pick up a copy of my latest book, *Critical Testing Processes*. If you've enjoyed this series of articles, it should be right up your alley.

Author Biography

Rex Black is President and Principal Consultant of RBCS, Inc. (www.RexBlackConsulting.com), a consultancy that provides testing experts world-wide, serving clients such as Bank One, Cisco, Hitachi, IMG, and Schlumberger in consulting, training, and hands-on implementation. He's written *Managing the Testing Process, Critical Testing Processes Volumes I* and *II*, and numerous articles, along with presenting papers and keynote speeches at international conferences.

Bibliography

R. Black, *Critical Testing Processes, Volume I*. Addison-Wesley, 2002.

E. Hendrickson, "More Testing, Worse Quality," www.qualitytree.com