Investing in Software Testing: High Fidelity Test Systems

Abstract

Realizing a solid return on your testing investment requires smart selection of tests. Cost of quality analysis tells us that it's cheaper to find and fix bugs before the customers do, but, to keep bugs away from customers, we have to find the ones that matter. Doing so requires that we understand how the customers will use the system.

Introduction

In the last article, I made a financial case, based on cost-of-quality analysis, for investing in software testing. However, just as smart stock market investing requires buying the *right* stocks, smart software testing involves carefully chosen tests. To achieve a positive return on the testing investment, testers must target this investment at building and applying the right test system. (I use the phrase *test system* to describe the test facilities, test environment, test data, test cases, and test execution processes.) The matter of what is the "right" test system is a multifaceted one, but let's start by looking in this article at the importance of a test system that truthfully replicates the customers' experiences of quality.

How to Waste Money on Software Testing

In the last article, the case study's return on investment arose from the pre-release detection (and repair) of "must-fix" bugs. A *must-fix* bug is an unacceptable defect that would at some point be identified (by users) and repaired (by the sustaining organization) over the course of the system lifecycle. By handling must-fix bugs before release, we leverage order-of-magnitude differences in the costs of nonconformance between internal and external failures.

However, you can test actions or features that few customers use, verify configurations no customer runs, and report problems no customer cares about. Because time and money are generally fixed during development projects, the effort spent—by testers, developers, and managers—on these pointless tests and bugs represents effort not spent on other tests and bugs that might be critical. To add insult to injury, the results of this misguided testing give management a false sense—either inflated or deflated—of system quality. Since testing is about both finding problems where the product is defective and increasing confidence where the product works, you can fairly say that a test team using the wrong test system is a poor investment of money and time. (For more details on this topic, see my upcoming book, *Critical Testing Processes, Volume I*.)

Test System Fidelity

I call a test system that allows testers to mimic customer usage a *high fidelity test system* because it faithfully reproduces the behaviors customers will experience when they use the system under test. The behaviors will either satisfy or dissatisfy the customers, leading to a positive or negative experience of product quality, respectively.

Let's look at a couple illustrations. In Figure 1, you see a situation where Customer A uses only a portion of the product. Through Test System A, Tester A uses that same portion and then some. Therefore, once testing is completed with Test System A, Tester A understands Customer A's experience of quality. Assuming the product coverage from Test System A represents the union of most customers' usage of the product, Test System A is a high fidelity test system. If bugs exist in the product that will plague the customer, Tester A will see them before the system ships. The programmers will have an opportunity to fix those bugs before release. And test manager can deliver an accurate, timely assessment of quality to the project management team, enabling smart decisions about release readiness and project progress.

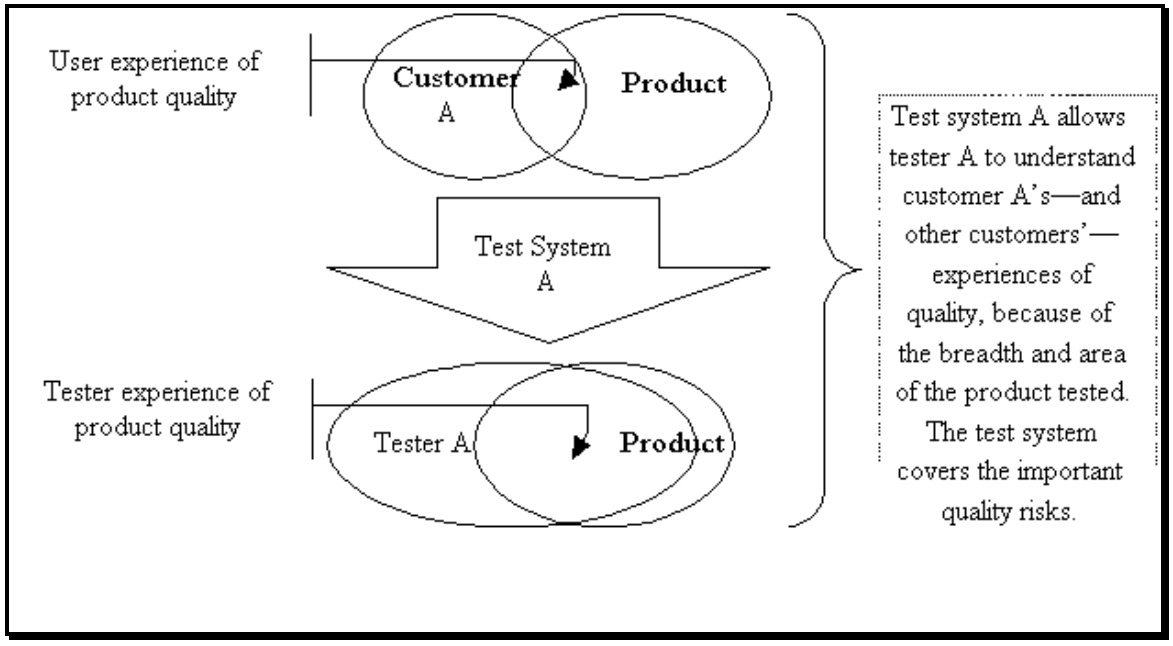

Figure 1: A High Fidelity Test System

Conversely, Figure 2 shows a low fidelity test system. Using Test System B, Tester B spends time testing a slender portion of the product, missing all the important pieces of functionality from Customer B's perspective. Because the coverage is so slight, it's likely that Test System B is a low fidelity test system when considered across the whole customer base. Testers using this test system will miss most must-fix bugs in the product, and therefore the programmers will not be able to repair the must-fix bugs before the system ships. Furthermore, the test results reported to the project management team will mislead them about the

state of system quality, leading to ill-informed decision-making and mistaken perceptions. All these factors contribute to a waste of the testing investment.

Let me sum up the distinction. High fidelity test systems focus on tests for key customer scenarios, in likely customer configurations, emphasizing problems that customers would consider important. In other words, testers mimic customer usage by applying high fidelity test systems. Low fidelity test systems test the wrong features, run on the wrong configurations, or report the wrong problems. Of course, it's important to remember that high fidelity and low fidelity aren't binary states. There's a spectrum from the perfect test system to the perfectly useless test system. A team of wise test professionals will work to develop a test system that, within schedule and budgetary constraints, has the highest fidelity possible.

A Cautionary Case Study

Once upon a time there was a test manager who managed a test team that had a fancy automated testing system. The system under test was a multi-OS, multi-database query and reporting system, and the test team had created a test system that sent canned queries and reports into the system under test, then automatically compared the results against baselines. They could test over a dozen OS/database combinations in a couple days and could run thousands of tests. Sound impressive? Well, these testers were wasting time and money.

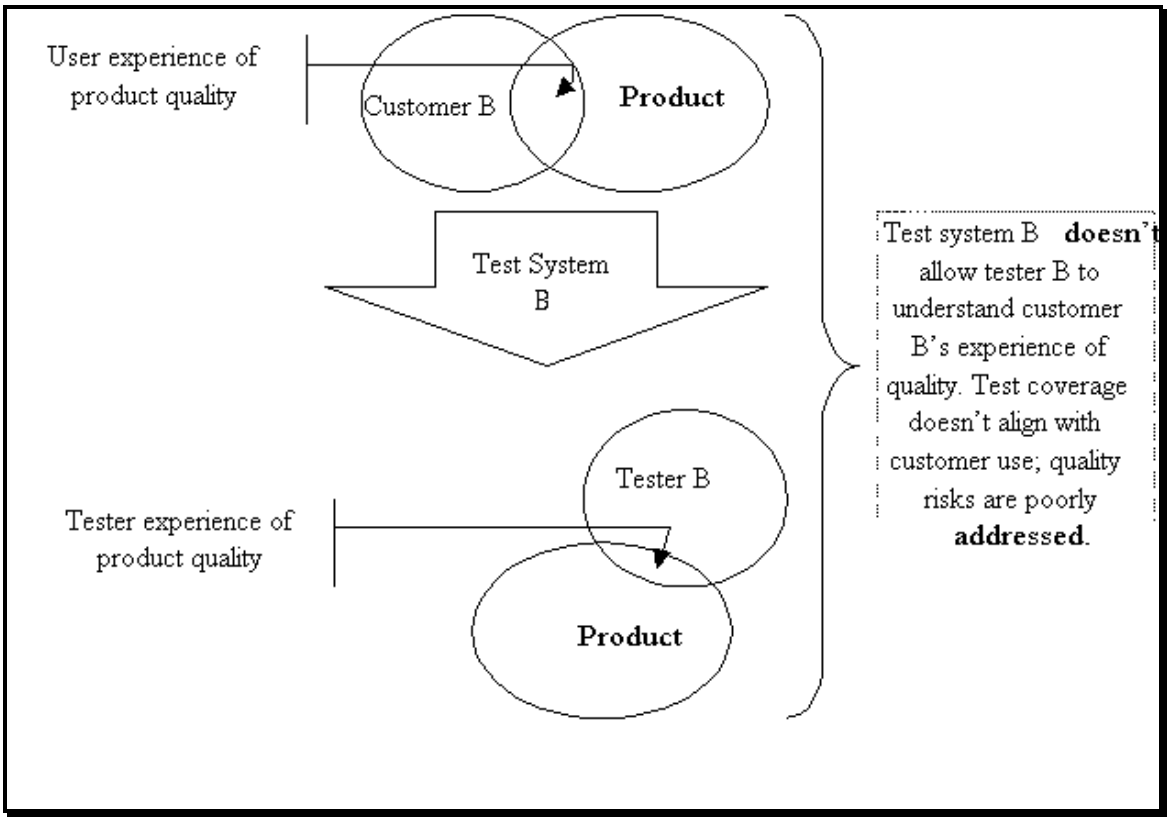

Figure 2: A Low Fidelity Test System

Why? The customer pain wasn't that the product tended to return incorrect query results. For the most part, that logic was solid and the test system rarely found bugs in one database or OS that didn't occur on all of them. Customers disliked the product because it was hard to install and because the ancillary and supporting tools and utilities didn't work. The test team, though, largely ignored these other aspects of the product and focused on the perpetuation and propagation of their automated test system across many different OS/database combinations, thus creating a low-fidelity test system that aligned poorly with customer usage.

The Next Step

The test manager in that case study was me, over a decade ago. From that mistake, I learned the importance of aligning testing with customer usage. I first started to think about the value of high fidelity test systems. This article has hopefully inspired you to think about that topic, too. Once you start to see the customer as the focal point of your test effort—the person whose experience of quality you are trying to predict before the product ships—you are on the first step of a journey that leads to intelligent management of quality risks.

However, while it is all well and good to assert that you should build high fidelity test systems, that is somewhat like telling someone that they must take an airplane to get from Austin to Aachen in one day. For experienced travelers, that suggestion is helpful. However, many people would remain confused about the journey and the destination. In the next article, I'll show you a couple analytical techniques you can use to establish a high fidelity test system.

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*, and numerous articles, along with presenting papers and keynote speeches at international conferences.

Bibliography

R. Black, *Critical Testing Processes*. Boston, Addison-Wesley, 2003.

R. Black, *Managing the Testing Process, 2nd Edition*. New York, Wiley, 2002.