Investing in Software Testing: The Cost of Software Quality

Abstract

Testing can be considered an investment. A software organization—whether an in-house IT shop, market-driven shrink-wrap software vendor, or Internet ASP—chooses to forego spending money on new projects or additional features to fund the test team. What's the return on that investment (ROI)? Cost of quality analysis provides one way to quantify ROI.

Introduction

The process of negotiating a software testing budget can be painful. Some project managers view testing as a necessary evil that occurs at the end of the project. In these people's minds, testing costs too much, takes too long, doesn't help them build the product, and can create hostility between the test team and the rest of the development organization. No wonder people who view testing this way spend as little as possible on it.

Other project managers, though, are inclined to spend more on testing. Why? Smart software managers understand that testing is an investment in quality. Out of the overall project budget, the project managers set aside some money for assessing the system and resolving the bugs that the testers find. Smart test managers have learned how to manage that investment wisely. In such circumstances, the test investment produces a positive return, fits within the overall project schedule, has quantifiable findings, and is seen as a definite contributor to the project. (For some ideas on helping to make your managers—and yourself—smarter see James Bullock's article, "Calculating the Value of Testing," and Rob Sabourin's article, "At Your Service," both available here at www.stickyminds.com.)

What Does Quality Cost?

The title of Phil Crosby's book says it all: *Quality Is Free*. Why is quality free? Like Crosby and J.M. Juran, Jim Campenella, in *Principles of Quality Costs*, illustrates a technique of analyzing the costs of quality start by breaking down these costs as follows:

$$C_{quality} = C_{conformance} + C_{nonconformance}$$

Conformance costs include prevention costs and appraisal costs. Prevention costs include money spent on quality assurance—tasks like training, requirements and code reviews, and other activities that promote good software. Appraisal costs include money spent on planning test activities, developing test cases and data, and executing those test cases once.

Nonconformance costs come in two flavors, internal failures and external failures. The costs of internal failure include all expenses that arise when test cases fail

the first time they're run, as they often do. A programmer incurs a cost of internal failure while debugging problems found during her own unit and component testing.

Once we get into formal testing in an independent test team, the costs of internal failure increase. Think through the process: The tester researches and reports the failure, the programmer finds and fixes the fault, the release engineer produces a new release, the system administration team installs that release in the test environment, and the tester retests the new release to confirm the fix and to check for regression.

The costs of external failure are those incurred when, rather than a tester finding a bug, the customer does. These costs will be even higher than those associated with either kind of internal failure, programmer-found or tester-found. In these cases, not only does the same process described for tester-found bugs occur, but you also incur the technical support overhead and the more expensive process of releasing a fix to the field rather than to the test lab. In addition, consider the intangible costs: Angry customers, damage to the company image, lost business, and maybe even lawsuits.

Two observations lay the foundation for the enlightened view of testing as an investment. First, like any cost equation in business, we will want to minimize the cost of quality. Second, while it is often cheaper to prevent problems than to repair them, if we must repair problems, internal failures cost less than external failures.

A Hypothetical Case Study

Let's use a hypothetical case study to illustrate the use of this cost of quality technique to analyze return on the testing investment. Suppose we have a software product in the field, with one new release every quarter. On average, each release contains 1,000 "must-fix" bugs—unacceptable defects—which we identify and repair over the life of the release. Currently, developers find and fix 250 of those bugs during development, while the customers find the rest. Suppose that you have analyzed the costs of internal and external failure. Bugs found by programmers costs $10 to fix. Bugs found by customers cost $1,000 to fix.

As shown in the "No Formal Testing" column in Figure 1, our cost of quality is three-quarters of a million dollars. It's not like this $750,000 expenditure is buying us anything, either. Given that 750 bugs escape to the field, it's a safe bet that customers are mad!

Suppose we calculate that bugs found by testers would cost $100 to fix. This is one-tenth what a bug costs if it escapes to our customers. So, we invest $70,000 per quarterly release in a manual testing process. The "Manual Testing" column shows how profitable this investment is. The testers find 350 bugs before the release, which cuts almost in half the number of bugs found by customers. This certainly will make the customers happier. This process improvement will also make the Chief Financial Officer happier, too: Our total cost of quality has

dropped to about half a million dollars and we enjoy a nice fat 350% return on our $70,000 investment.

In some cases, we can do even better. For example, suppose that we invest $150,000 in test automation tools. Let's assume we intend to recapture a return on that investment across the next twelve quarterly releases. Would we be happy if that investment in test automation helped us find about 40% more bugs? Finding 500 bugs in the test process would lower the overall customer bug find count for each release to 250. Certainly, customers would be much happier to have the more-thoroughly tested system. In addition, cost of quality would fall to a little under $400,000, a 445% return on investment.

	A	B	C	D
1	**Testing Investment Options: ROI Analysis**			
2				
3		**No Formal**	**Manual**	**Automated**
4	**Testing**	**Testing**	**Testing**	**Testing**
5	Staff	$0	$60,000	$60,000
6	Infrastructure	0	10,000	10,000
7	Tools	0	0	12,500
8	**Total Investment**	0	70,000	82,500
9				
10	**Development**			
11	Must-Fix Bugs Found	250	250	250
12	Fix Cost (Internal Failure)	2,500	2,500	2,500
13				
14	**Testing**			
15	Must-Fix Bugs Found	0	350	500
16	Fix Cost (Internal Failure)	0	35,000	50,000
17				
18	**Customer Support**			
19	Must-Fix Bugs Reported	750	400	250
20	Fix Cost (External Failure)	750,000	400,000	250,000
21				
22	**Cost of Quality**			
23	Conformance	**$0**	**$70,000**	**$82,500**
24	Nonconformance	**$752,500**	**$437,500**	**$302,500**
25	**Total CoQ**	**$752,500**	**$507,500**	**$385,000**
26				
27	**Return on Investment**	**#N/A**	**350%**	**445%**

Figure 1: Using Cost of Quality to Analyze Return on Investment

This is a huge improvement over the initial situation. We are realizing a quantifiable and substantial return on our testing investment. We are also making our customers happier.

Is This For Real and How Do We Get There?

Cost of quality analyses on software process improvement bear out these figures. In *Principles of Quality Costs*, Campenella presents a case study from Raytheon that describes reductions in the cost of software quality from a whopping seventy percent of the total production cost to twenty to thirty percent. To put these percentages in more concrete terms, suppose you currently develop, deploy, and support systems at an average cost of $1,000,000 each. Reductions in the cost of software quality like Raytheon achieved would reduce this average cost to around $500,000. Can your organization use an extra $500,000 per system? While testing is only part of the investment in quality, it is an important part. Certainly a substantial investment is justifiable to achieve such phenomenal gains.

To get started, you'll need a management team wise enough to look at the cost of quality over the entire life of the software release. A management team that ignores the long term and focuses just on the budget required to get the software out the door initially does not see testing as an investment. Quality is given lip-service when the only priorities are shipping something—anything—on a given schedule and within a given budget.

You'll also want to get a better grip on what quality actually costs your organization. As Crosby points out in *Quality Is Free*, most organizations that haven't looked at quality costs carefully significantly underestimate those costs. Campenella describes rigorous accounting techniques in *Principles of Quality Costs*. However, first-order estimates—provided you are conservative—can be used to build a solid, credible business case. Ask programmers, technical support personnel, testers, business analysts, and managers (development, technical support, help desk, IT, sales, and marketing) what percentage of their time they spend dealing with each kind of failure, internal and external. To turn these percentages into dollars—or euros, or pounds, or yen, or whatever—figure out what a staff-hour costs in your organization. For example, $100 per hour is typical in the United States for technical organizations, including costs like health insurance, facilities, and other overheads beyond just salary. Add in other costs as appropriate. For example, in one company I talked to, their response to field failures was to fly a programmer to the client's site, along with sufficient tools to fix the bug, and keep him there until the problem was fixed, which was typically about a week. Last minute airfare, hotel costs, meals, and car rental added about $2,000 to the $4,000 cost associated with the programmer's lost time. (For additional ideas on doing this analysis, see my book, *Managing the Testing Process, Second Edition*, Mark Fewster and Dorothy Graham's book, *Software Test Automation*, or Johanna Rothman's web site, www.jrothman.com.) Your management team will probably be surprised at what quality costs your organization right now, and how little value it's getting from that expenditure.

However, having supportive, far-sighted management is only a necessary—not a sufficient—condition for achieving positive returns on your test investment. Just as in the stock market, there are right and wrong ways to invest. Picking the right tests, managing the appropriate quality risks, using the proper tools and techniques, and driving testing throughout the organization will result in optimal returns, while failure in any one of these areas can mean disappointing or even negative returns. In the next few columns, I'll explore each of these topics.

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, *Managing the Testing Process, Second Edition*. Wiley, New York, 2002.

J. Bullock, Calculating the Value of Testing, *Software Testing and Quality Engineering*, Volume 2, Issue 3, 56-62 (May/June, 2000).

P. Crosby, *Quality is Free*. McGraw-Hill, New York, 1979.

J. Campanella, ed., *Principles of Quality Costs*. ASQ Quality Press, Milwaukee, 1999.

M. Fewster and D. Graham, *Software Test Automation*. Harlow, Addison-Wesley, 1999.

S. Kan, *Metrics and Models in Software Quality Engineering*. Addison-Wesley, Reading, 1995.