RBCS
TIME TESTED.
TESTING IMPROVED.
www.RBCS-US.com

What IT Managers Should Know about Testing: How to Analyze the Return on the Testing Investment

Payoff Idea

Project managers must develop quality systems that provide the needed features in a timely and cost-effective fashion. Testing systems prior to release is necessary to assess their quality, but what's the return on that testing investment? This article describes four quantifiable metrics for testing ROI.

Introduction

To run a successful project, system development managers must constantly ask four questions:

1. Features: Can we deliver a system with the functions the users and customers need?
2. Schedule: Can we deliver a system in a timely fashion?
3. Budget: Can we deliver a system that is profitable, yielding more value than it cost to build?
4. Quality: Can we deliver a system that has all the right quality characteristics, such as capability, security, accuracy, reliability, usability, and performance?

Using configuration management techniques, managers can answer the first question. Using traditional project management techniques like work-breakdown-structures, earned value analysis, project accounting, and cost-benefit analysis, managers can answer the second and third questions.

Using the latest tools and techniques in system testing, test professionals can provide accurate, timely assessments of system quality that help managers answer the fourth question. However, testing a medium to large system can consume a significant portion of the project budget. So, smart system development managers should ask, "What quantifiable value do I receive from this large investment in testing?"

In this article, you'll learn how to answer that question. There are four quantifiable values that testing provides. I'll give you techniques for estimating these values. Using these estimates, you'll be ready to make hard-headed, dollars-and-cents decisions about how much testing you should conduct. I'll illustrate these ideas with a running case study.

Assessment of Quality

Quality guru J.M. Juran's defined quality as "fitness for use," meaning the predominant presence of attributes and behaviors that satisfy users, customers, and other system stakeholders and the relative absence of attributes and behaviors that would dissatisfy those same stakeholders.[1]

For software, the attributes and behaviors that influence the users' and customers' experiences of quality vary, but often include characteristics such as security, functionality, performance, usability, operability, and so forth.[2] [3]

During the development or maintenance of a system, these quality characteristics are at risk. In other words, there is some possibility of undesirable behaviors in the system. Testing allows project managers to have a realistic assessment of the current levels of system quality during the project and prior to release. This information allows project managers to intelligently manage these risks to system quality and guide the project to success.[4]

The levels of risk are more acute for some system quality characteristics than for others. Therefore, smart test professionals use quality risk analysis techniques to focus the testing where the risks are highest.

Elevated levels of risk in certain areas result from both technical and business considerations. For e-commerce systems, performance is a key technical risk, because of the design of the Internet, how people connect to it, and the relative immaturity of Internet technology. Performance is also a key business risk, since customers are always a click away from the competition should they decide that a particular site is too slow to complete their transaction.[5] [6]

Cost of Quality

The first value of testing lives in the relative costs of good and bad quality. We can start analyzing these costs by breaking the *cost of quality* into two components:

---

1 Juran, J.M. *Juran on Planning for Quality*. New York: Free Press, 1988.

2 ISO 9126-1:2000, *Information Technology: Software Product Quality, International Organization for Standardization*.

3 Kan, Stephan Kan. *Metrics and Models in Software Quality Engineering, 2e.* Reading, MA: Addison-Wesley, 2003.

4 Black, Rex. *Critical Testing Processes*. Boston, MA: Addison-Wesley, 2003.

5 Black, Rex. *Managing the Testing Process, 2e.* New York: John Wiley and Sons, 2002.

6 Craig, Rick and Stefan Jaskiel. *Systematic Software Testing*. New York: Artech House, 2002.

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

Equation 1: Cost of Quality

Costs of conformance are the costs of building quality into the system. The costs of quality assurance activities like requirements reviews and process-improvement initiatives are costs of conformance, referred to as *costs of prevention*. Some of the costs of testing a system are also costs of conformance costs, specifically *costs of detection*.

Some of the costs of testing occur when testing successfully locates bugs. In these cases, some of the bugs indicate serious technical or business-related problems with system quality. The project team should fix such serious bugs before they deliver the system. Fixing bugs changes the system, which invalidates some or all of the testing previously carried out. So, we need to retest (or *regression test*) the system.

The costs of fixing bugs and retesting the system are *costs of nonconformance*. Specifically, they are *costs of internal failure*. The system failed, but we detected the problem prior to delivery, and fixed it.

Alternatively, if testing fails to detect some serious problems, or if project managers decide to ignore such problems and deliver the system anyway, the costs of dealing with the bugs in the field are costs of nonconformance, called *costs of external failure*.[7]

Time and again, when people have studied quality costs, two general observations have held. First, it's cheaper to build things right the first time than to fix them later. Second, the cost of a failure tends to increase significantly the later in the development process it is discovered.

An example of the value of doing it right the first time comes from Raytheon Electronic Systems. In the course of implementing software process improvements based on the Capability Maturity Model, they reduced the overall cost of quality from about 70% of project cost to around 20 to 30%. To put this in concrete terms, if you implemented such changes with the same level of success, you could carry out a $1,000,000 project for around $500,000.[8]

Examples of the tendency of bug costs to balloon over time have been around since Barry Boehm's paper "Software Engineering" published in 1976. It showed order-of-magnitude increases in failure costs as bugs moved from requirements

[7] Juran, J.M., and Frank Gryna, editors. *Quality Control Handbook.* New York: McGraw-Hill, 1988.

[8] Campanella, Jack, ed. *Principles of Quality Costs*. Milwaukee: ASQ Quality Press, 1999.

to design to coding to unit test to system test.[9] A study two decades later by Walter Baziuk showed similar increases, with some bugs found in the field costing as much as 2,900 times more than bugs found in requirements.[10] It stands to reason: A simple mis-statement in a requirements specification can turn into two or three bad decisions in system design which can turn into ten or twenty bugs in the system.

So here's one benefit of testing. If a bug costs $100 to find and fix during testing, and $1,000 to find and fix in the field, then testing yields a bounty of $900 per bug found and fixed. As long as the investment in testing—the detection costs of conformance—are less than $900 per bug found, we have a positive return on investment based on cost of quality analysis alone.

Activity-based costing can provide precise figures for costs of quality. Many of us work in environments where such data are not readily available, though. How do we estimate the costs of detection, the costs of internal failure, and the costs of external failure?

Start with the total budget of your testing team. For any expenditures that result in reusable test assets—setting up test environments, buying test tools, developing automated or manual test scripts, etc.—deduct the value of those assets that will be realized on future projects. Also, subtract costs associated with regression testing of fixed builds prior to release. This gives you the costs of detection, your investment in testing.

Now, for the costs of internal failure, start with the costs associated with regression testing that you calculated in the previous step. Now, estimate the percentage effort spent by the rest of the development organization in fixing bugs. You might find you can adequately approximate this figure by using the proportion of the non-testing project budget that is expended after development delivers the first feature-complete release for testing. Dividing this cost by the number of bugs found during testing gives you your cost of internal failure per bug.

Finally, for the costs of external failure, consider the costs both of regular maintenance releases and emergency patches. What percentage of their time do developers, testers, help desk or technical support, configuration management, and others spend on such efforts? Are there any extraordinary costs associated with fixing bugs in the field? For example, do you have service level agreements or other guarantees with clients, customers, or users? Do technical personnel

[9] Barry Boehm, "Software Engineering." IEEE Transactions on Computer Software Engineering, Volume 1, Issue 4, pages 1226-1241.

[10] Walter Baziuk, "BNR/NORTEL Path to Improve Product Quality, Reliability, and Customer Satisfaction," presented at the Sixth International Symposium on Software Reliability Engineering, Toulouse, France, October 1995.

have to travel to repair problems on-site? What are the prospects of disgruntled customers or lost business as a result of bugs? Might your company be sued or even held criminally liable in the event of a field failure? All of these costs, divided by the number of bugs found after release of a system, give you your cost of external failure per bug. (You can also analyze the comparative costs of internal and external failure dynamically to determine appropriate release dates, as described in a recent presentation.)[11]

Case Study: Cost of Quality

Let me illustrate this technique and the cost of quality return on the testing investment based on an example from one of our clients. We spent about nine months on a testing project, of which the first six months were test asset development and the last three months were test execution. The project budget was about $1,000,000. Due to reuse of the test assets and the fact that most of the staff-hours were expended during test execution, the net conformance costs of detection were about $400,000.

Detection Costs	
Test Budget	$1,000,000
Future Value of Assets Created	100,000
Regression Test Costs	500,000
Net Detection Costs	**$400,000**
Must-Fix Bugs Found During Testing	1,500
Approximate Cost per Bug	**267**

Table 1: Conformance Costs of Detection

We found about fifteen hundred must-fix defects during testing. During the testing period, a team of about 30 developers worked to fix the bugs. Conservatively estimating a fully burdened staff cost of $100,000 per year, at least three-quarters of a million dollars were spent fixing bugs. I add to that our regression test costs (in Table 1) to calculate the nonconformance costs of internal failure at $1.25 million.

[11] Tim Koomen, "Damage Prevented: The Value of Testing Measured," presented at the STAR East 2003 conference, Orlando, Florida, May 2003.

<u>**Internal Failure Costs**</u>	
Pre-Release Bug Fix Costs	$750,000
Regression Test Costs	500,000
Net Internal Failure Costs	**$1,250,000**
Must-Fix Bugs Found During Testing	1,500
Approximate Cost per Bug	**833**

Table 2: Nonconformance Costs of Internal Failure

For the first six months after product release, roughly 30 developers, 10 testers, and 20 technical support staff worked to sustain the product. Conservatively estimating that only 50% of that sustaining effort was due to fixing bugs—as opposed to adding new features—I calculate a nonconformance cost of external failure at $1.5 million. During this six-month period, about 500 must-fix bugs were discovered.

<u>**External Failure Costs**</u>	
Sustaining Costs	$3,000,000
Percentage Bug-Related	50%
Net External Failure Costs	**$1,500,000**
Must-Fix Bug Found After Testing	500
Approximate Cost per Bug	**3,000**

Table 3: Nonconformance Costs of External Failure

Now I can put this all together to calculate return on the testing investment. I can calculate the total cost of quality using Equation 1. To calculate the return on the testing investment based on cost of quality, notice that I can figure net benefit of testing based on Table 2 and Table 3. Since we had 2,000 must-fix bugs total, we would have faced $6 million in quality costs had all those bugs been discovered as external failures.

The net benefit of testing is therefore this potentially higher cost of quality minus the actual cost of quality, almost $3 million. Dividing by the conformance costs of detection shown in Table 1—which is the amount invested in testing—I obtain a return on investment of about 700%. This figure is consistent with another recent study for the US Department of Defense.[12]

Return on Investment	
Cost of Quality (w/testing)	$3,150,000
Cost of Quality (w/o testing)	6,000,000
Net Benefit of Testing	**$2,850,000**
Approximate Testing ROI (CoQ)	**713%**

Table 4: Return on Testing Investment

The Value of Known Bugs

The cost of quality model is useful in estimating the return on the testing investment, but it underestimates the return. For one thing, cost of quality only assigns value to bugs that get fixed prior to release.

However, some bugs should not be fixed before release. The costs (in terms of schedule as well as budget) and the risks (in terms of possible regression) of fixing some bugs can outweigh the benefits.

Even if we don't fix all the bugs we find during testing, we enjoy three benefits to having found out about the bugs before release.

1. In some cases, we can prevent the user from seeing the bug.
2. We can warn users in the release notes so they can avoid the bug.
3. We can provide a workaround and other information to the help disk or technical support staff.

For example, suppose we have a bug that affects older versions of the Internet Explorer. We could add a check that prevents the users from accessing our site until they've downloaded a newer version. We could perhaps warn users in the release notes that using older versions of the Internet Explorer may result in particular operations not working properly. Or, technical support could advise customers who call in about this known bug to download a free update to Internet Explorer.

It's difficult to quantify the value for the first two benefits, but the value of the third benefit is not so hard to estimate.

Having provided a workaround for a bug to technical support, we can assume that they will resolve calls related to this bug more quickly. Rather than spending

12 David Rico, "Using Cost Benefit Analyses to Develop Software Process Improvement Strategies," Data and Analysis for Software, September 2000.

time troubleshooting the problem, the technical support person can quickly look up the bug and advise the user of a workaround. Not only does that save money, but it also results in a happier customer. The only effort required is the time to transfer the knowledge, which will be minimal if the test team careful documents its test cases and bug reports.

Case Study: The Value of Known Bugs

After the three month test execution period for our client, prior to widespread customer release of the system, two members of the test team spent two weeks working with the technical support team to transfer knowledge about what we had tested, what worked, and what bugs remained. The cost of the four person-weeks of tester time amounted to less than one percent of the test budget.

About 500 known, significant bugs remained in the product. The technical support manager estimated that each support call would cost about $6, though a support call for a known bug would cost 50% less. Marketing estimated 50,000 customers for the product in the first year. Assuming conservatively that 100 of these customers would complain about each of these bugs, that's still 50,000 support calls, one per customer.

Bugs Not Fixed But Known	
Known, Significant, Unfixed Bugs	500
Cost Per Support Call	$6
Support Calls per Bug	100
Cost Savings for Known Bugs	50%
Net Savings From Known Bugs	**$150,000**

Table 5: Value of Known Bugs Found by Testing

You might notice something interesting in Table 5. We can calculate the support cost of each bug at $600. This figure is well below the cost of internal failure for must-fix bugs shown in Table 2, which supports the project management decision to ship without fixing these bugs. Based on purely financial considerations, we should not fix any bug that would cost more to fix before release than would be saved in the long run. (Of course, sometimes considerations other than financial ones—such as ethical or legal considerations—must influence the fix/defer decisions as well.) The value of testing for these bugs accrues not from having fixed them, but from having merely found them.

Factoring in this additional return on the testing investment, the net benefit of testing is now $3 million dollars. The return on investment has reached 750%.

Return on Investment	
Cost of Quality (w/testing)	$3,150,000
Cost of Quality (w/o testing)	6,000,000
Known Bug Savings	150,000
Net Benefit of Testing	**$3,000,000**
Approximate Testing ROI	**750%**

Table 6: Revised Return on Testing Investment

The Value of Risk Mitigation

We've counted the return on the testing investment in two ways, but there're still some values missing from our analysis. For our third value, notice that we've not counted any value from tests that pass.

In the earlier section on assessing quality, I mentioned that various risks to system quality exist. In personal and business life, we often handle risks by buying insurance.

Testing is a form of proactive insurance for systems. If we test systems prior to release, those tests that pass give us confidence in the features, functions, and behaviors that those tests cover. Those tests that fail give us an opportunity to avoid unanticipated external failure costs later. In economic terms, we can substitute the before-the-fact risk mitigation provided by testing for the after-the-fact reimbursement for losses provided by insurance.

In the insurance business, the starting point for calculating the cost of an insurance policy is called the expected payout:

$$EP = \sum_{1}^{n} (p(l_i) \times \bar{c}(l_i))$$

Equation 2: Expected Payout for an Insurance Policy

The expected payout is the sum, across all the covered risks, of the probability of a loss times the average cost of a loss.

The expected payout is the cost to the insurance company to sell a policy. Therefore, it's the lowest possible amount we could pay for a policy that covers us against some set of risks. Typically, we'd have to pay operating costs and profits on top of that.

So, the expected payout for all the quality risks covered by our test effort is another value added by testing.

Case Study: The Value of Risk Mitigation

In our case study project, we identified significant risks to the quality of the system and developed 175 test cases. When run for the first time, over 50% of the test cases failed. When a test case failed, it identified on average one must-fix bug.

From the analysis shown in Table 3, we know that the loss to the company would average $3,000 for each unidentified bug delivered. Multiplying each loss times the probability for all 175 tests, we see that the testing delivered $262,500 in value in the form of insurance substitute.

Tests that Pass	
Number of Test Cases	175
Percentage Tests Failed First Attempt	50%
Average Must-Fix Bugs Per Failed Test	1
Cost per Bug Shipped to Field	$3,000
Insurance Value (Expected Payout)	**$262,500**

Table 7: Expected Payout Value of Testing

Factoring in this additional return on the testing investment, the net benefit of testing is now over $3.25 million. The return on investment exceeds 800%.

Return on Investment	
Cost of Quality (w/testing)	$3,150,000
Cost of Quality (w/o testing)	6,000,000
Known Bug Savings	150,000
Insurance Value	262,500
Net Benefit of Testing	**$3,262,500**
Approximate Testing ROI	**816%**

Table 8: Revised Return on the Testing Investment

The Value of Good Advice

Even with this significant return on investment identified, we're still missing one last, major, quantifiable value of testing. For our fourth value, notice that we have not yet counted the value of the information produced by testing. However, project managers must have this information to make the smart project decisions.

Capers Jones has identified four main factors in project failure. One is poor project tracking. Based on these four factors plus project size, he reports risks of project failure ranging from 2% for small projects to 85% for the largest projects.

On a medium-sized project, the risk is 40% based on poor project tracking alone. If all four factors are handled well, the risk for a medium-sized project drops to 20%.[13]

If a system development project fails, we have lost at least the entire project budget. After all, this money is now gone. Failed projects usually recover pennies on the dollar for their capital assets and the intellectual property created by the project.

(In many cases what is gone, too, is the opportunity the project was poised to capture. For example, Apple was one of the first-to-market with a personal digital assistant (PDA), the Newton. However, one of the early purchasers of the Newton was the cartoonist Garry Trudeau. He mocked the poor quality of the handwriting recognition feature in his *Doonesbury* cartoon strips. Since these cartoons were popular with the same demographic groups in the United States to whom Apple was marketing the Newton, this was the worst kind of negative publicity at the worst possible time. Given the enormously lucrative nature of the PDA market now, surely Apple lost far more than just the sunk costs of the now-defunct Newton.)

What percentage of good project tracking can we attribute to good test result reporting? In my experience, the effective communication of test status is essential, especially at the end of the project when the risk of serious budget or scheduled overrun or outright project failure is the highest. "How is the quality of the system?" is one of the four questions project managers must answer. So, a simple, reasonable estimate is that 25% of the value of good project tracking arises from a good assessment of quality from the test team.

Case Study: The Value of Good Advice

For our final look at the case study project, the overall project budget was about $4 million. This was a medium-sized, nine-month development project. So, good project tracking reduced the risk of project failure by 20%, worth $800,000. The test information contributed $200,000 of that value.

[13] Jones, T. Capers. *Estimating Software Costs*. New York: McGraw-Hill, 1995.

<u>**Good Project Tracking**</u>	
Development Budget	$3,000,000
Testing Budget	1,000,000
Total Project Budget	**$4,000,000**
Reduced Risk Due to Project Tracking	20%
Value of Good Project Tracking	800,000
Net Value of Test Information	**$200,000**

Table 9: Value of Test Information

So, the total net benefit of testing for this project was almost $3.5 million. We provided our client with an over eight-fold return on their $400,000 investment in testing.

<u>**Return on Investment**</u>	
Cost of Quality (w/testing)	$3,150,000
Cost of Quality (w/o testing)	6,000,000
Known Bug Savings	150,000
Insurance Value	262,500
Test Information	200,000
Net Benefit of Testing	**$3,462,500**
Approximate Testing ROI	**866%**

Table 10: Final Return on the Testing Investment

Conclusion

In this article, I explained four solid, quantifiable benefits from testing:

1. Find bugs that get fixed—or even prevent them.
2. Find bugs that don't get fixed, but are known.
3. Run tests that mitigate (potentially expensive) risks.
4. Guide the project with timely, accurate, credible information.

In the case study, I demonstrated a return on the testing investment of over 850%.

Of course, these kinds of returns require good management of the testing process. We must focus testing on the key risks to obtain benefits in each of these areas. We must extract meaningful metrics from our test results to guide the project. We must perform testing in an effective and efficient fashion, even on projects that involve off-shore or outsource development teams.

The ability to achieve this kind of testing requires trained, skilled test professionals. Management must ensure that the right test team is in place.

In the next four articles in this series, I'll demonstrate how system development managers can carry out these important activities.

About the Author

With a quarter-century of software and systems engineering experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. As the leader of RBCS, Rex is the most prolific author practicing in the field of software testing today. His popular first book, *Managing the Testing Process*, has sold over 30,000 copies around the world, including Japanese, Chinese, and Indian releases. His three other books on testing, *Critical Testing Processes*, *Foundations of Software Testing*, and *Pragmatic Software Testing*, have also sold thousands of copies, including Hebrew, Indian, Chinese, Japanese and Russian editions. He has written over twenty-five articles, presented hundreds of papers, workshops, and seminars, and given about thirty keynote speeches at conferences and events around the world. Rex is the President of both the International Software Testing Qualifications Board and the American Software Testing Qualifications Board.