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

Four Ideas for Improving Test Efficiency

Nothing good lasts forever. We have entered another economic downturn, and no one seems to know how long it will last. For the foreseeable future, management will exhort testers and test teams to do more with less. A tedious refrain, indeed, but you can improve your chances of weathering this economic storm if you take steps now to address this efficiency fixation.

In this brief article, I'll give you four ideas you can implement to improve test efficiency. All can show results quickly, within the next six months. Better yet, none require sizeable investments which you could never talk your managers into making in this current economic situation. By achieving quick, measurable improvements, you will position yourself as a stalwart supporter of the larger organizational cost-cutting goals, always smart in a down economy.

Know Your Efficiency

The first idea—and the foundation for the others—is that you should know your efficiency to know what to improve. All too often, test teams have unclear goals. Without clear goals, how can you measure your efficiency? Efficiency at what? Cost per what?

Here are three common goals for test teams:

- Find bugs
- Reduce risk
- Build confidence

You should work with your stakeholders—not just the people on the project, but others in the organization who rely on testing—to determine the right goals for your team.

With the goals established, ask yourself, can you measure your efficiency in each area? What is the average cost of detecting and repairing a bug found by your test team, and how does that compare with the cost of a bug found in production? (I describe this method of measuring test efficiency in detail in my article, "Testing ROI: What IT Managers Should Know," found in the Basic Library at www.rbcs-us.com.) What risks do you cover in your testing, and how much does it cost on average to cover each risk? What requirements, use cases, user stories, or other specification elements do you cover in your testing, and how much does it cost on average to cover each element? Only by knowing your team's efficiency can you hope to improve it.

Institute Risk-Based Testing

I mentioned risk reduction as a key testing goal. Many people agree, but few people can speak objectively about how they serve this goal. However, those people who have instituted analytical risk-based testing strategies can.

Let me be clear on what I mean by analytical risk-based testing. Risk is the possibility of a negative or undesirable outcome, so a quality risk is a possible way that something about your organization's products or services could negatively affect customer, user, or stakeholder satisfaction. Through testing, we can reduce the overall level of quality risk. Analytical risk-based testing uses an analysis of quality risks to prioritize tests and allocate testing effort. We involve key technical *and* business stakeholders in this process.

Risk-based testing provides a number of efficiency benefits:

- You find the most important bugs earlier in test execution, reducing risk of schedule delay.
- You find more important bugs than unimportant bugs, reducing the time spent chasing trivialities.
- You provide the option of reducing the test execution period in the event of a schedule crunch without accepting unduly high risks.

You can learn more about how to implement risk-based testing in Chapter 3 of my recent book, *Advanced Software Testing: Volume II.* You can also read the article I co-wrote with an RBCS client, CA, on our experiences with piloting risk-based testing at one of their locations. This article is featured in the January 2009 issue of Better Software magazine.

Tighten Up Your Test Set

With many of our clients, RBCS assessments reveal that they are dragging around heavy, unnecessarily-large regression test sets. Once a test is written, it goes into the regression test set, never to be removed. However, in the absence of complete test automation, this leads to inefficient, prolonged test execution periods. The scope of the regression test work will increase with each new feature, each bug fix, each patch, eventually overwhelming the team.

Once you have instituted risk-based testing, you can establish traceability between risks and test cases, identifying those risks which you are over-testing. You can then remove or consolidate certain tests.

You can also apply fundamental test design principles to do identify redundant tests. We had one client that, after taking our Test Engineering Foundation course, applied the ideas in that course to reduce the regression test set from 800 test cases to 300 test cases. Since regression testing made up most of the test execution effort for this team, you can imagine the kind of immediate efficiency gain that occurred.

Introduce Lightweight Test Automation

I mentioned complete test automation above. That's sometimes seen as an easy way to improve test efficiency. However, for many of our clients, that approach proves chimerical. The return on the test automation investment some of our clients see is low, zero, or even negative. Even when the return is strongly positive, for many traditional forms of GUI-based test automation, the payback period is too far in the future and the initial investment is too high.

However, there are cheap, lightweight approaches to test automation. We helped one of our clients, Arrowhead Electronic Healthcare (www.aheh.com), create a test automation tool called a *dumb monkey*. It was designed and implemented using open source tools, so the tool budget was zero. It required a total of 120 person-hours to create. Within four months, it had already saved almost three times that much in testing effort. For more information, see the article I co-wrote with our client, which you can find in the January 2009 issue of Software Test and Performance magazine.

Conclusion

In this article, I've shown you four ideas you can implement quickly to improve your efficiency. Start by clearly defining your team's goals, then derive efficiency metrics for those goals and measure your team now. With that baseline measurement, move on to put risk-based testing in place, ensuring the right focus for your effort. Next, apply risk-based testing and other test fundamentals to reduce the overall size of your test set while not increasing the level of regression risk on release. Finally, use dumb monkeys and other lightweight test automation tools to tackle manual, repetitive test tasks, saving your people for other more creative tasks. With these changes in place, measure your efficiency again six months or a year from now. If you are like most of our clients, you'll have some sizeable improvements to show off for your managers.

Author Bio

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. 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. 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 35,000 copies around the world, including Japanese, Chinese, and Indian releases. His five other books on testing, *Advanced Software Testing: Volume I*, *Advanced Software Testing: Volume II, Critical Testing Processes*, *Foundations of Software Testing*, and *Pragmatic Software Testing*, have also sold tens of thousands of copies, including Hebrew, Indian, Chinese, Japanese and Russian editions. He has written over thirty 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 the International Software Testing Qualifications Board and a Director of the American Software Testing Qualifications Board.