

Two Weeks to Better Testing in 2008

We asked some clients, associates, and colleagues what some of their top New Year's resolutions were. While we can't share them all, you won't be surprised to find one of the top "to dos" for 2008 is to find practical ways to improve testing. So, if you're like most testers, you are time constrained and need to make improvements quickly that show fast results. Here I present three practical ideas which you can put into action in just two weeks, which will make a noticeable difference to get your testing year started off right.

Get Hip to Risk-Based Testing

I have a simple rule of thumb for test execution: Find the scary stuff first. How do we do this? Make smart guesses about where high-impact bugs are likely. How do we do that? Risk-based testing.

In a nutshell, risk-based testing consists of the following:

1. Identify specific risks to system quality.
2. Assess and assign the level of risk for each risk, based on likelihood (technical considerations) and impact (business considerations).
3. Allocate test effort and prioritize (sequence) test execution based on risk.
4. Revise the risk analysis at regular intervals in the project, including after testing the first build.

You can make this process as formal or as informal as necessary. I have helped clients get started doing risk-based testing in as little as one day, though one week is more typical. For more ideas on how, see my article, "Quality Risk Analysis," in the Library at www.rbc-us.com, or my books *Managing the Testing Process* (for the test management perspective) or *Pragmatic Software Testing* (for the test analyst perspective).

Whip Those Bug Reports into Shape

One of the major deliverables for us as testers is the bug report. But, like Rodney Dangerfield, the bug report gets "no respect" in too many organizations. Just because we write them all the time doesn't mean they aren't critical – quite the contrary – and it doesn't mean we know how to write them well. Most test groups have opportunities to improve their bug reporting process.

When I do test assessments for clients, I always look at the quality of the bug reports. I focus on three questions:

1. What is the percentage of rejected bug reports?
2. What is the percentage of duplicate bug reports?
3. Do all project stakeholder groups feel they are getting the information they need from the bug reports?

If the answer to questions one or two is, “More than 5%,” I do further analysis as to why. (Hint: This isn’t always a matter of tester competence, so don’t assume it is.) If the answer to question three is, “No,” then I spend time figuring out which project stakeholders are being overlooked or underserved. Recommendations in my assessment report will include ways to get these measures where they ought to be. Asking the stakeholders what they need from the bug reports is a great way to start – and to improve your relationships with your coworkers, too.

Read a Book on Testing

Most practicing testers have never read a book on testing. This is regrettable. We have a lot we can learn from each other in this field, but we have to reach out to gain that knowledge.

(Lest you consider this suggestion self-serving, let me point out that writing technical books yields meager book royalties. In fact, on an hourly basis it’s more lucrative to work as a bagger at a grocery store. Other benefits, including the opportunity to improve our field, are what motivate most of us.)

There are many good books on testing out there now. Here’s a very small selection:

What You Want

General tips and techniques for test engineers

Object-oriented testing

Web testing

Security testing

Dynamic test strategies and techniques

Books to Read

Pragmatic Software Testing, Rex Black
A Practitioner’s Guide to Software Test Design, Lee Copeland

Testing Object-Oriented Systems, Robert Binder

The Web Testing Handbook, Steve Splaine

Testing Web Security, Steve Splaine
How to Break Software Security, James Whittaker

T-Map Next, Tim Koomen et al
How to Break Software, James Whittaker

What You Want

Books to Read

Test management

Managing the Testing Process, Rex Black
Systematic Software Testing, Rick Craig

Test process assessment
and improvement

Critical Testing Processes, Rex Black
Test Process Improvement, Martin Pol et al

ISTQB tester certification

Foundations of Software Testing, Rex Black et al
The Testing Practitioner, ed. Erik van Veenendaal

I have read each of these books (some of which I also wrote or co-wrote). I can promise you that, if you need to learn about the topic in the left column of the table, reading one of the books in the right column will repay you in hours and hours saved over the years, as well as teaching you at least one or two good ideas you can put in place immediately.