

2009

PACIFIC NW SOFTWARE
QUALITY
CONFERENCE



MOVING
QUALITY
FORWARD

OCTOBER 27-28, 2009

Conference Paper Excerpt

From the

CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Half-Baked Ideas for Rapid Test Management

Jon Bach

jbtestpilot@hotmail.com

Abstract

As a test manager, there are stakeholders to report to and direct reports to collect reports from. There are processes and practices, principles and procedures, politics and protocols – all of which might bind you in some way from being creative. With all of that, “Moving Quality Forward” might mean you have to take some risk and break some rules.

This paper is about my experience as a test manager for one year at LexisNexis, leading 15 people on 4 projects. It’s about how I tried to find time to make the projects better by making my people better despite having no time for formal training.

It’s about the ideas I tried as a Rapid Test Manager focusing on the development of testing skill in each tester while testing was structured using heuristics and just-in-time documentation, rather than through detailed procedures and test case management systems. It’s about how I tried to move quality forward and what happened next.

Biography

I graduated from college as a journalist, but never pursued a career in it. I worked as a fireman, dishwasher, bookstore clerk, and even wrote a book about me and my Dad. But in 1996, I tried testing. At first, I didn't believe I could be very useful as a tester, because I wasn't a programmer.

My brother James Bach, who had been testing for 14 years at the time, said testing was about learning, but I didn't believe him. He said I needed to embrace complexity, and at that, I gave him no argument. He said my success as a tester would be determined by the speed and enthusiasm with which I learned about the products being tested. He said I needed is to be fearless about learning. He made it sound like a noble quest, then he gave me something to test.

Fourteen years later, I'm still here because it's a lot like journalism. In that time, I've been a keynote speaker at international testing conferences, I've co-authored a Microsoft Patterns & Practices book on Acceptance Testing, and I've led 15 people on 4 projects simultaneously. What follows is about that most recent job.

Copyright 2009, Jon Bach

A New Hope

I landed at LexisNexis last year – ready for my biggest challenge – leading 15 people on 4 different projects. Here, I was going to conquer the world with new ways to test and lead testers. I was going to experiment and have a lot of fun finally having a chance to manage a big group of testers the way I wanted to.

One project was in maintenance mode. Two others were agile-like, with testers working in two-week and six week sprints respectively. The fourth project was waterfall, sporting a long release schedule and formal test cycles.

For my staff, I was their third test manager that year.

A few of the guys were new to the team, but most had been grinding for a while. So, they were tired, and no one had much training as software testers.

I wanted to introduce this team to Rapid Testing, which is a testing methodology James developed over the last 20 years, in conjunction with Cem Kaner, Michael Bolton, and a little help from me. Rapid Testing focuses on the development of testing skill in each tester and the testing itself is structured using heuristics and concise documentation, rather than through ponderously detailed procedures or test case management systems. It does not consist of a set of approved test techniques or templates. Rather, Rapid Testing encompasses any useful test technique. There are no best practices, only best practices *in context*.

To manage the team well, I felt that I needed to do a few things quickly: establish my credibility as a tester and a leader, learn the technology under test, learn how the testers worked, and establish a mechanism for understanding testing status each day.

Here are some of the things I came up with. I call them "half-baked ideas" because I had no idea how they'd work or how they'd be received. They were literally ideas of ingredients that I put into the oven of a software project in my attempt to develop (bake) them.

What follows is an account of how they came out of the oven.

Family Feud

The first thing I wanted to do, like a lot of managers, is get to know my team. I held meetings with my crew, one by one, to get a feel for what was on their minds and what they might be concerned about. Based on what I heard, I made up a survey and sent it around.

The main purpose was to be sure I was hearing and understanding the important issues that that were on the minds of the testers. The survey included open-ended questions such as "What are 3 aspects of your ideal manager?" and "What one project problem do you

wished was solved NOW?" It also included some questions that were more closed and somewhat less serious, such as "Name the most interesting project codename we've used." and "Name a customer that tends to be mentioned a lot in meetings."

I decided to send a mail with the results, then stopped. Is that the most creative way I could introduce myself to the team? By sending out a simple email of results? Too boring.

So what popped into my head was the "survey says" game show, Family Feud. A hundred people surveyed about everyday objects, events, and choices. Contestants have to guess the results of the survey. So, I put on a suit and tie and played emcee for the game.

Result: Success. After I revealed the results, I took off the tie and told them what I was going to do to follow up on their answers. I don't know if anybody got anything out of it, but almost a year later, everyone remembers it. It was also fun because 6 months later, I sent out the survey results to remind them what their answers were to see how they'd changed. I was pleased to discover that the one project problem everyone wished was solved now (the slowness of Team Foundation Server), was solved.

Open-Book Testing

When I was a tester on Microsoft Flight Simulator in 2002, I had an idea. What if I could create a set of questions for myself that I did NOT know the answers to? And what if I let those questions guide my exploration to find new bugs?

I asked the testers to create a set of quiz questions about each of the products they were testing. Then I answered the questions by working with those products. In some cases, I hooked up a projector and answered the questions in real-time while my staff watched me flounder. That way they could see me learning. It was basically an open-book test.

The exam they created had questions such as:

- What is the difference between tags, notes, and issues?
- How can you blank an entire database?
- Where can you change field properties?
- How can you tell who is in the database at the same time as you?

Open-book testing is a great way to teach new testers about a particular technology. You don't tell them what it is and how to use it. Instead, ask them questions that force them to find out for themselves. As testers learn, they also may find bugs.

Result: Success. By putting myself through that process, I seemed to earn credibility with the team. When new testers came on board, they used the open book questions to guide their exploration. They also seemed to like coming up with new questions to ask themselves.

Dawn Patrol

A Dawn Patrol is a group testing event held before normal working hours – 6 AM, to be exact. We did one a week for four weeks. We picked a product to test and each tester tried to find bugs in it. There was a lot of chatter and questions going back and forth.

My role was to take notes. My screen was projected on a wall so they could see what I wrote. Apart from observing how they tested, I was introducing them to the idea of note-taking and responding to scrutiny, asking them why and how they did certain tests.

Managing Rapid Testing requires a very hands-on approach, at least some of the time. I think testing is best managed by getting directly involved in the work. I wanted my testers to know that I was a serious tester. I also needed to model for them the behavior I expected from a professional tester, so I got involved in the testing, too – right then and there, but also because I needed to learn all I could about the products they were testing.

Result: We never did them again. I'm not sure why. I got too busy, no one took the lead, it was too early in the morning, maybe all of these things. But it did build teamwork that lasted for my tenure there. I established myself as both a learner and a teacher, and showed how exploration was something they already knew how to do, but they could get better at it if they practiced.

This is an example of the "Training Wheels" effect in process improvement. A team that tries a new process may later drop it because they have learned something or changed in some way that makes the process no longer necessary. That's how training wheels work. Some organizations cling to procedures and paperwork that long ago lost relevance. But, in Rapid Testing, the point is to regularly ask ourselves if a simpler process might not work better.

Status Reporting Experiments

I had my staff send me written status reports at the end of the day so I could keep track of what they were all doing. They initially helped me get a feel for the rhythm of work on each of the test projects. The reports consisted of two sections: highlights and blocking issues. But as I gained a feel for the team, I stopped reading them and began to manage more by walking around. I did not tell them to stop producing them, however, because I wanted an archive. I wanted the option to read them weeks later, like cracking open a black box if there were to be a crash.

I also experimented with getting status by having the testers post index cards with testing tasks on their cube walls. This was an Agile-style dashboard so they didn't have to be interrupted when I came by their cube. I had testers post index cards on their walls showing the testing tasks in which they were currently engaged. That helped me understand the status of the team at any moment, just by walking around. But the idea

failed during the first Snow Day in December 2008 when LexisNexis wound up closing for the day. As people worked from home, I had no way to know their status.

But luckily, one of the testers realized that ScrumWorksPro would enable them to do the same thing online. So, now we use that. In other words, we found it was a worthwhile process, we just took it virtual.

Color-Aided Test Design

Some half-baked ideas came from the testers. Well, McDonald's, sort of.

McDonald's was making their yearly Monopoly gamepiece promotion, and I was collecting the pieces and posting them on my cube wall. Knowing I love analogies from life about testing, one of my testers dropped by to talk about rapid test design and he challenged me to say what Monopoly had to do with that. Thinking fast, I suggested that the different colored properties in Monopoly could be compared to different levels of test importance – Baltic for low priority, Boardwalk for high.

I then challenged him to do something with that idea. He decided to try color-coding his testing spreadsheet in the same way the Monopoly colors were represented on the board so that we could see, at a glance, the different kinds of tests on it.

But it turns out he didn't have an interest in it as much as he thought. He said it took a little too much work, and I believed him, though I didn't sit down with him to analyze why. He said it was hard to keep up with every new test he created and put in the spreadsheet. It was also hard to see the colors in a huge spreadsheet at a glance, so the purpose was moot.

Maybe someday I'll try it myself, perhaps on an Agile project with a Big Visible Chart of stories with colors denoting priority?

Testing Hybrids

Imagine a test case – a set of steps you follow to get an expected result. Now imagine a scenario – a vague mission statement or workflow that's up to you to invent to achieve an expected result. Hybrids were partially documented test procedures -- not detailed step-by-step procedures, but also not a typical exploratory testing charter. It is a checklist, but it is a checklist of ideas of things to see and do, like a trip itinerary.

It's another idea that came from the team. I was telling them that we could test better and faster if we ignored thick test procedure documentation. But one of the testers felt that the hybrid approach could give me the structure I wanted while cutting way down on paperwork.

Rapid Testing works well this way. By asking "What testing is needed, here and now? How do we prepare ourselves for testing that we'll need to do tomorrow? Are we

delivering the information our clients need?" again and again, every day, perhaps several times a day. In Rapid Testing, you are either a test leader, or you are training to become a test leader. I believe that testing is a challenging intellectual investigation, and Rapid Testing is about getting the most out of people to meet that challenge.

That's why encouraging testers to try experiments within (or without) a test process is important. They try things, they learn from them, and they come to own the process for themselves. This gives them confidence and makes them more committed to their work.

Here are some other ideas, quickly baked and out of the oven:

Lightning Talks – Book your team in a conference room for one hour with a projector and a timer. But three days prior, give them time to volunteer to deliver a 5-minute talk on anything project-related that they think the team could benefit from knowing – a tool demo (like using the free Windows Media Encoder to capture video) , a trick (like pressing F5 in Notepad to imprint an instant timestamp), or a website (like the wonderfully useful www.testingreflections.com).

Tester Show and Tell – this is like the lightning talks, only longer. This can happen in email, too. Ask your staff their favorite things to show off. What are their favorite Windows shortcuts (like holding down the Windows logo key and pressing E to launch Windows Explorer), what's their favorite testing website, what tools do they have loaded on their machine that are really useful? What the best bug they ever found on their current project? Give them a chance to show off or to simply recommend something and you may find a lot of "a ha" moments. That's what happened to me. I learned about Windows Media Encoder and <Windows key>+L to lock Windows, and in exchange, I showed them PerlClip.

Using WME and VLC for videos – Give them an assignment to record a 5-minute video of a test they did or something useful to the team. They might end up using it to record a short screencast of the next bug they find and attach the WMV file to the bug report. Both are free tools, and the ensuing recording can be put in a training archive. It gives your testers a chance to practice story-telling and reporting, as well as refine their exploration techniques. LiveMeeting also allows screencasts to be recorded and stored for future playback.

Dogfooding – (comes from "eating our own dogfood" – a way to use the product you're testing as if you were a customer yourself) -- Is there a way you can use your product you're testing to accomplish a task on your project? For example , if you're shipping a database-based product, can you test it by creating a database of all of your tests? If it's a web-based wiki dashboard app of some kind, can you use it to track project status updates? One of my testers used his product CaseMap – a tool for helping lawyers organize facts, witnesses, and evidence -- to organize his test cases and found

Conclusion

When I join a test team, I have my favorite ways of working, like everybody. I am biased toward exploratory testing and session-based test management. I'm skeptical of thick documentation, grandiose test automation, and heavy process. Even Agile projects don't seem agile enough for me. There is a lot of discipline required and not all technical professionals can work like that.

As a test manager, I think it's important to float ideas out there without over-testing them first. I take a service approach to leadership, and I'm excited to solve a problem fast, even though it's not all the way thought through. I'd rather trade time for the fun of experimentation. The risk, of course, is that haste makes waste. A half-baked idea can come out of the oven doughy and mushy. But it can also be "good enough" for its purpose.

Good enough does not mean mediocre. It means, "stopping the search for alternatives due to the cost of finding a more optimal solution." The word "good" means quality (or value) and the word "enough" means stop. If further time spent trying to improve an idea is too costly, take it out of the oven the way it is and see what happens.

In other words, look at management situations not in terms of a machine that must operate in some rigorous and inhuman way, nor a math problem to be cleverly solved. See yourself as a tester trying to move the quality of your team forward in little ways. Your half-baked idea may just be the training wheels you needed to get past the given hurdle or make your point.