Dinner with a Rapid Test Manager

By James and Jonathan Bach

"The first thing I did was to learn what their day was like," he told me. "Did the test process give them freedom to think? Or was the work of investigation obstructed by a smothering blanket of pre-planned test cases?"

We were interrupted briefly by the waiter taking our order. Then my brother resumed his narrative. "And of course, as the new manager, they needed to test me, too."

Jon Bach sat across from me as I took notes. He was telling me about his test group at LexisNexis. In a bit of role reversal, I was interviewing him about his test management methods.

Jon graduated from school as a journalist, but never pursued a career in it. He worked as a fireman, dishwasher, bookstore clerk, he even wrote a book. Then, in '96, I convinced him to try testing. At first, he didn't believe he could be useful as a tester, because he wasn't a programmer. Don't testers have to be computer scientists? No, I told him. What you need is to be fearless about learning. You must embrace complexity. Success as a tester is determined by the speed and enthusiasm with which you learn about the products you test.

I trained Jon, and as I predicted, he got in at Microsoft and soon distinguished himself as a tester and test leader. He gained experience in several groups there, worked for my company for a while, then went back to the Seattle area, eventually landing at LexisNexis. Today he's respected in the Context-Driven testing community as a test manager of sparkling creativity.

The reason I dragged him to dinner, that evening, was what he told me on the phone. He spoke of dawn patrols, Family Feud games, and reverse open-book testing. He mentioned color-coded test strategy, drive-by test coaching, and index card test status dashboards. These are not ordinary test management practices. So, I invoked older brother privilege and demanded a brain dump.

"The team I inherited consisted of fifteen people on four diverse projects." Jon told me. "One project was in maintenance mode. Two others were agile-like, with testers working in two-week and six week sprints respectively. While the fourth project was a waterfall classic, sporting a long release schedule and formal test cycles. 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."

Jon wanted to introduce his team to Rapid Testing, which is the testing methodology I have developed over the last twenty years, in conjunction with Cem Kaner, Michael Bolton, and Jon himself. 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," we say.

Rapid testing is a fundamentally sapient process. That means, though we use tools, good people are essential to success. To cultivate that team of good people, Jon needed to do a few things quickly: establish his 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 his more unorthodox tactics for doing those things.

Family Feud

"I held meetings with the testers, 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 of Jon's survey was to be sure he 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."

Jon used a creative way to announce the results of the survey. "We played Family Feud," he said. "You know, that game where they survey a hundred people about everyday objects, events, and choices. Then the contestants have to guess the results of the survey." So, he put on a suit and tie and played emcee for the game. After he'd revealed the results, he took off the tie and told them what he was going to do to follow up on their answers.

Open-Book Testing and Dawn Patrols

"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 while they watched. That way they could see me learning. It's basically an open-book test."

The quiz 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?

The point of an open-book quiz like this is not to test knowledge, but to drive learning and develop resourcefulness. 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. By putting himself through that process, Jon earned credibility with the team.

"The Dawn Patrols also helped me learn the product, but they were more about getting the testers to learn about a more professional approach to exploratory testing." Jon told me as I tried to eat salad

right-handed while taking notes. "A Dawn Patrol is a group testing event held before normal working hours. We did one a week for four weeks. We picked a product to test and each tester tried to find bugs in it. There's a lot of crosstalk and questions going back and forth. My role was to take notes. My computer 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 notetaking and responding to scrutiny."

Managing Rapid Testing requires a very hands-on approach, at least some of the time. We think testing is best managed by getting directly involved in the work. Jon wanted his testers to know that he was a true tester, himself. He also needed to model for them the behavior he expected from a professional tester. And of course, he needed to learn all he could about the products they were testing. Open-book testing and dawn patrols are a good way to get that done.

Status Reporting Experiments

"I had the testers do daily written status reports for a while, but stopped after a couple of months. I also experimented with getting status by having the testers post index cards with testing tasks on their cube walls. We don't do that anymore, either."

Daily status written status reports initially helped Jon 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 he gained a feel for the team, he began to manage more by walking around. He also attended the stand-up meetings for the two Scrum projects. After a couple of months, it became clear that daily status reports were no longer needed.

I call this 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-- unable to shed training wheels, they slow themselves down. But, in Rapid Testing, we regularly ask ourselves if a simpler process might not work better.

The index cards is a little different case. Jon had testers post index cards on their walls showing the testing tasks they were currently engaged in. That helped him understand the status of the team at any moment, just by walking around. But one of the testers realized that ScrumWorksPro would enable them to do the same thing online. So, now they use that. In other words, they found it was a worthwhile process, they just took it virtual.

"Let's Try That"

I asked him what process improvements had been suggested by the testers.

"Color-coded test matrixes." He said immediately. "The idea came from McDonald's, which was making its yearly Monopoly promotion, and I was collecting the pieces and posting them on my cube wall. One of the testers dropped by to talk about rapid test design and he challenged me to say what Monopoly had to do with testing. Thinking fast, I suggested that since the different colored properties in Monopoly indicate their value, we could also indicate different values that tests might have by using color. I challenged him to do something with that idea. He decided to try color-coding his testing spreadsheet so that we could see, at a glance, the different kinds of tests on it."

"Are you still doing that?" I asked.

"Nope. It turns out the way he did it took a little too much work. It was hard to keep up with it. We have to find an easier way to use color. But I'm glad we tried it."

"Okay. Name an innovation that did stick."

"Hmm. Hybrid test cases."

"What are those?" I'd never heard of that.

"That's a kind of partially documented test procedure. It's not a detailed step-by-step thing, but also not a typical exploratory testing charter. It's more like a loose checklist of things to see and do. It's another idea that came from the team. I was telling them that we could test better and faster if we stopped using thick test procedure documentation. But one of the testers felt that the hybrid approach could give him the structure he wanted while cutting way down on paperwork. I think it's working for him."

Rapid Testing works partly 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?" These questions are asked again and again. Every day. Several times a day. Who asks them? Whomever is leading the testing. But in Rapid Testing, you are either a test leader, or you are training to become a test leader. Jon and 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 with the test process is so 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.

What's the common thread?

When Jon joins a test team, he does have his own favorite ways of working. He is biased toward exploratory testing and session-based test management. He is skeptical of thick documentation and grandiose test automation. But that's not what's most interesting about his style of leadership.

The thing that strikes me most is that he takes a service approach to leadership. He listens before he directs. He submits himself to the judgment of his team to earn their trust. He wants each tester to be more powerful and marketable as a result of working for him. Jon looks at a testing situation not in terms of a machine that must operate in some rigorous and inhuman way. Nor is it some math problem to be cleverly solved. Instead he sees a team of people, with a variety of backgrounds, skills, and talents, helping each other understand the state of the product.

To test well, there is no substitute for testers who learn fast, have good ideas, and keep their eyes and minds opens. In our experience, a small number of testers, properly led and trained, can outperform a legion of script drones.