Pair Testing: How I Brought Developers into the Test Lab

by Jonathan Kohl

AS A BLACK BOX TESTER WORKING ON A variety of development projects, I was often approached by developers who were trying out test-driven development and wanted to learn more about testing. To show them how I worked. I involved them in pair testing, a technique in which two people test an application at the same computer. Interestingly enough, the developers taught me as much as I taught them.

Let me explain how it worked. Before a pair testing session, the developer and I would meet to determine the focus and scope of the test. We would pick an area of the program to test and establish our goal. Sometimes the goal would be to track down an elusive bug, sometimes to ensure that customer acceptance test criteria were met, or sometimes to find bugs in a new piece of functionality. We would write our goals and testing ideas on a whiteboard, keeping a copy of the finished notes for our own use.

Later, the developer would join me at the computer. He would watch how my exploratory testing techniques led to new ideas and plans of attack. Soon, he too would be actively testing. As the sessions progressed, we would revisit our goals and come up with new ideas for testing.

Software testing professional Jonathan Kohl found that while showing developers how he tests software, he learned a great deal about the application itself.

Mutual Learning

The benefits of pair testing extend beyond the developer and tester involved. Developers learn how to test their own code more effectively and gain a new perspective on how their software might be used. Testers gain a more thorough understanding of the application they are testing and learn debugging techniques to find causes of defects. Both developers and testers learn how to uncover more information to write more effective defect reports. Additionally, pair testing can break down communication barriers between developers and testers and facilitate team building.

During pair testing sessions, developers regularly told me that they had never thought of looking at software testing the way I did. The "what would happen if I tried this…" mindset was not as intuitive to them. They tended to think of tests that reflected a typical way of using the software; I thought of tests that might cause failures. As I applied my usual techniques they typically would say something such as, "Stop! Slow down! What are you doing?" Then I would stop and explain my reasoning. The types of bounds conditions or input validation tests that I immediately tried on input fields of new features surprised them. They were also surprised by my attempts to get around the control flow of the program.

The developers weren't the only ones learning something new. I was often struck by how developers' perspectives and testing techniques were different from my own. I tend to focus on the user's perspective, while developers tend to focus on what the program tells them about what is going on behind the scenes. As they shared their knowledge of the underlying code, I learned what areas of the product might contain weaknesses and gained application-specific knowledge that helped me track down problems more effectively and provided more information when reporting a defect.

Once developers applied what they learned about testing to their development projects, they came up with great testing scenarios on their own. Best of all, it became difficult to find defects in the developer's code after pair testing.

Info to Go

- By exploring software with testers, developers learn how their software might be used.
- By looking at software through the eyes of developers, testers can learn more about the application itself.

Collaborative Problem Solving

Pair testing can also be used as a way to track down hard-to-reproduce defects. Once, after several hours of testing an application, I uncovered a defect that occurred only sporadically. I suspected clients had reported this defect in the form of several seemingly unrelated is-

RINUS BORGSTEEDE/CORBIS SABA

sues. To track it down, I pair tested with a senior developer. While he had a suspicion of what was causing the defect, we could not repeat it when testing together. At the end of that session, he installed a debugger on my machine and told me what information needed to be captured if it occurred again.

The next time the defect occurred, we tested together until we were confident we had narrowed down the cause of the problem. He knew what code was problematic, so we brainstormed related integration tests to narrow down where the problem might occur on the code level. From that list, I derived several more tests. In almost every case, the defect occurred. We were able to track down the problem, fix it, and develop some solid unit tests for the fix based on our sample of test cases.

The whole process took only a couple of days. In my experience as a tester, tracking down difficult defects on my own has taken much longer. Through collaboration, we found the cause more quickly and were able to ship the software with a high level of confidence.

Working Together in New Ways

Sometimes, instead of black box testing together at a machine, we collaborated on automated unit test ideas. We began by working on test cases that involved permutations and combinations. From the large number of possibilities, we determined a sample size and chose a set of test cases for automated unit test development. In addition to these tests, I suggested other testing ideas for developers to implement when writing their automated unit tests. When I tested the same feature, I complemented the developers' unit tests with scenarios they had not covered. Some developers were much more comfortable with this type of collaboration than with pair testing together at a computer in the testing lab.

Risks and Limitations

Pair testing is not a cure-all. As with any technique, there are risks involved, and some pair testing efforts fail. Some failures can be traced to what Agile QA Manager Janet Gregory calls a lack of trust between the developer and the tester. "If one or the other goes in with

the idea that it is a one-way learning experience, the experience will fail." Pair testing is only effective in an environment of mutual respect and trust.

In pair programming, both people already understand program design and architecture. In testing, the developer might not understand the testing focus. Whoever is "driving" during pair testing must ensure that the other party is actively participating and understands what is going on. Encourage thinking and talking aloud, keeping the other person informed on the motivation behind your actions.

Teaming a tester who has a knack for black box testing and finding defects with someone who is intimately familiar with the underlying code is a great combination. Try it. **{end}**

Getting Started

To use pair testing in your team, start with these steps. While this is not an exhaustive list, it will help:

1. Choose a developer you trust and who buys into the concept of pair testing.
2. Pick a suitably-sized project. Don't try to test the whole application in one session. Pair testing works well when testing new functionality and when both participants have been working on the project since inception.
3. Plan up front. Determine a time to test, the length of the session (an hour or so is a good place to start), a break schedule, and a testing focus definition. Clarify the goals, and define the outputs of the test sessions. Computer engineer and software tester Javan Gargus notes that in programming there is generally only one kind of output: documented code. In testing, there can be several outputs: defect reports, test documentation, test cases, etc. You should define the intended outputs of pair testing efforts so that both parties understand what is required.
4. Use an environment that is suitable for two people to test together at one machine. Be sure you can work without interruptions and are free to talk to each other.
5. Evaluate outcomes. How successful were the pair testing sessions? What would you do differently next time?

Jonathan Kohl (jonathan@kohl.ca) is a software testing professional in Calgary, Alberta, Canada. *He thanks Javan Gargus, Janet Gregory, and Elizabeth Kohl for their help with this article.*

Sticky Notes

For more on the following topics go to www.stickyminds.com/bettersoftware

- Pair testing
- Agile development & testing