Repeating the Unrepeatable Bug

by Jonathan Kohl

WHEN I WAS A NEW RECRUIT AT A SOFTware company, I stumbled across an application crash. I tried to repeat it by replicating the steps I had taken to get there, but I couldn't. I had saved a screen shot and the error log, so I asked some of the other testers about the bug. They told me it was an "unrepeatable bug," and that each one of them had found it at some point. While I spent a couple of hours that afternoon working on it, in the end I told the testing team that I couldn't repeat it either. The team repeated the mantra that we often use in these situations: "If we don't have a repeatable case, we can't fix it." Without a repeatable case, I was just a voice in the dark.

When it became clear that this sporadic, unrepeatable bug was negatively impacting customers, the team decided that we should spend time looking into it. The next time I found it, I paired with a programmer and watched how he attacked the problem. The first thing I noticed was that he wasn't interested in the details of what I had been doing when the bug appeared. Instead, he looked at the big picture, envisioning the entire system in his mind, along with the intricate interactions within the system. I asked him to explain what he was thinking, to show me the clues he saw in stack traces, and what he had done in the program code to help trap the error the next time it occurred.

Eventually, we were able to repeat the bug regularly at a layer behind the GUI. However, because we couldn't always repeat it at the GUI layer, it took some time to convince the team that we had tracked down the cause. The developer and I spent time building our case. We followed hunches and tried out different theories. When we felt our case was solid, we presented our findings. We got a green light to fix the bug and, later, were satisfied to learn that we had eradicated a sporadic bug, which our customers would no longer have to deal with.

Jonathan Kohl says there is no such thing as an unrepeatable bug.

RINUS BORGSTEEDE/REDUX PLUS

So-called "unrepeatable bugs" have always bothered me, especially sporadic, high-impact bugs, such as application crashes, data corruption, and memory leaks. Often, what seems like a sporadic problem in the test lab may be a constant problem for a customer.

When I was starting out as a tester, I worked on unrepeatable bugs during lulls between release cycles. Through a combination of luck and stubbornness, I learned that I could often find repeatable cases for these bugs. Since then, I have honed my testing skills by finding repeatable cases for high-impact unrepeatable bugs. Here's what I've learned along the way.

Getting a Repeatable Case

There isn't a set formula to repeating an unrepeatable case. It takes a combination of activities to narrow down the cause. When you find a high-impact unrepeatable bug, continue to gather evidence as you perform other testing tasks.

Whenever you see the bug, save all the information you can. Create a special folder and save stack traces, screen shots, notes, and anything else that seems relevant to the case. Revisit the folder and review the data, so that when you work with developers on these kinds of bugs, you have information to help them troubleshoot. They often will recognize patterns or clues that you missed. In your spare time, check the bug database for similar bugs. I have noticed that several seemingly unrelated unrepeatable bugs have identical stack traces in the bug reports. Investigation revealed that they were all related to the same problem.

Instead of focusing on details, start by finding patterns. It is often more important to look at the big picture than to look at the steps you took to get there. Watch for patterns among the different sightings of the bug. When talking to others, ask questions and watch for patterns in their answers. If you get a hunch about a particular pattern, ask if anyone noticed that pattern. They may have in-

sight that didn't seem important when they logged the bug.

In one instance, an atypically long delay between my actions was the first pattern that emerged. As I was talking with another tester, the idea came to me. I asked him if, when he saw the bug, time had elapsed from one action to another. He wasn't sure, but I had a hunch to go on. I followed the hunch and tested with a stopwatch. I discovered that waiting more than a certain period of time between two actions would always repeat the bug. Now that we had an initial pattern with which to work, the developers were able to repeat the bug and find the source.

Remember that the GUI may not be the best place to repeat all bugs. Many unrepeatable bugs are caused by conditions deep in the program code or in the third-party software that the application uses. Usually, a problem condition in the code is tripped in the back end, and as you exercise the application through the GUI, it is hanging around, waiting to go off. In many cases, the bugs can consistently be reproduced when using a testable interface behind the GUI. For example, with a Web application, data that is changed in the GUI isn't always updated in the database. You may not be able to tell that anything is wrong until much later when something else in the application manipulates the incorrectly unchanged data, causing a sporadic crash at the user interface layer. However, if you test behind the GUI, it's easier to tell what is happening to the data.

Use automated testing tools to help track down unrepeatable bugs. Automated tests can quickly simulate conditions that are difficult for a single tester to repeat. Sporadic errors can be repeated more often when running certain tests quickly with a tool, and test scripts can be developed when a scenario seems to cause the bug more frequently.

Trust Your Instinct

Follow hunches, think outside the box, and try testing techniques or possible fixes even if they seem unorthodox. As Web server tuning analyst Justin Domshy says, "I've learned to be willing to try anything for a potential fix. Sometimes the strangest sounding solutions turn out to be the right ones." Both of us have seen Web applications fail due to third-party database drivers and application server failures. In one case, removing an unrelated software program from an application server caused the Web application errors to stop. Reinstalling it revealed that a database driver was the culprit.

If you find a repeatable case for a sporadic bug, don't be surprised if no one believes you at first. These kinds of bugs don't fit a typical development cycle, so people may react to them in unexpected ways. Be diplomatic and continue to gather evidence. Collaborating with others is important, partly because those who help you reach conclusions about bugs will champion the repeatable case. Developers can help point out why the patterns emerge the way they do and can help identify other testable interfaces to repeat the bug.

Develop a catalog of failures and their causes as you go. You can draw on this experience when you see similar problems. Learn the weak spots of programming languages used. For example, I have spent a lot of time tracking down pointer errors in applications developed in languages such as C and C++. Be sure to find out what third-party software is being used in the system. Find out what servers, drivers, APIs, and other tools are relied on by the application. Many times third-party tools are the cause of unrepeatable bugs. To identify potential weak spots, it's a good idea to create a model of the system of the application you are testing.

Be humble, courteous, and respectful of others' decisions. The cost of fixing many low-impact unrepeatable bugs may outweigh the benefits. However, if you believe that a high-impact bug is being set aside because it doesn't have a repeatable case, don't be discouraged. Remember, there is no such thing as an unrepeatable bug. Stick to your guns and try thinking about the bug in a different way. **{end}**

Jonathan Kohl (jonathan@kohl.ca) is a software testing consultant with Kohl Concepts Inc. in Calgary, Alberta, Canada. Check out his other writings at www.kohl.ca.