Ghost Bug Busters

by Karen Johnson

TESTERS FOCUS ON BUGS: WE FIND A BUG, we repeat a sequence of steps to reproduce it, and we lobby to have the bug fixed. But not all bugs let us follow this simple path. The nasty bugs, some of the juiciest, aren't easy to replicate. I think of these bugs as ghosts—things we've seen but cannot conjure up again. These bugs leave us haunted with doubts about a system.

But true software bugs aren't random and they don't go away. They reappear only when you follow the same steps, use the same data, and work with the system in the same state. And while we may have forgotten the steps we took, or failed to notice other system factors, the bug remains in the code. However, ghosts have origins; they have a reason and a condition that will bring them back. For this, you need a strategy to raise them up again. Here are some tips to help you track down your ghosts.

Record What You Can

You may be tempted to start replicating a bug the moment you notice it. But don't. Instead, stop and record all the information you can before you close an error message or restart a workstation. Take a few moments to ask yourself: What other applications are running? How much memory is available? Is the database server still up and running? Was a network connection lost? What information can you find in the application error log or database transaction log?

You should also take note of other dialog boxes in the application that may be displayed. Be sure to record the exact error message and note the dll being reported if the error is a GPF (general protection fault). Use the print screen keys if you can and save a snapshot of the application in trouble; the snapshot may provide important clues to your investigation.

QUICK LOOK

- 7 tips for replicating difficult defects
- Advice on when to ask for help...and when not to

Get Quiet Quick

If you're having trouble replicating a bug, eliminate all the distractions in your workspace. Become quiet; turn your thoughts inward. If you need space, ask for it. Now...think *carefully:* Where have you been in the application? What data did you use? What steps did you perform? What's different about the steps you ran this time?

I worked in one department where if a developer passed by and saw an error message on a test workstation, he'd immediately get involved by asking questions. While I appreciated good follow-up and code ownership, sometimes the questions would come so quickly after the error that it was more of a distraction than a benefit. So I learned to shut the monitor off and allow myself time to think and replicate the bug before being confronted with well-intentioned questions.

Question the Background

Was a background service running that could have affected the system? For example, when working with NT there can be scheduled NT service or jobs running periodically in the background (with or without notification to the desktop). Was the network connection disrupted? If a client application you're testing requires network connection and the connection is interrupted, the client application may encounter a system error. Was the database connection lost? If so, the application can throw an error. Was the transaction log in the database full—causing what might appear to be a sporadic defect? Any one of these background checks may reveal the origin of your ghost.

One system I tested had a client application and an NT service running on a PC workstation. On one occasion, while I was working with the client application, a crash occurred. I spent days trying to replicate the bug with no success. Then I decided to set up the NT service to provide screen notification when the job ran. When the NT service ran, a GPF in the application occurred due to a temporary low-memory situation. Poof! My ghost was caught. Never again have I assumed that the background couldn't have an impact in my testing.

Check the Build You're Testing

Sometimes you don't try to replicate a bug until days after you first see it. You could be working with a different build by the time you get around to it. Could the bug have become resolved or been pushed elsewhere in the code? Be aware of the build changes in your test environment, because they could explain your disappearing ghosts.

Write When You Test

Writing can help you with difficult bugs in significant ways. By keeping a journal as you work, you have notes to use when you need to recreate a scenario. Make it a practice to record every step you make in exact order *as you test*, and write the steps in an abbreviated form so your writing isn't a distraction to your testing.

But even if you didn't record information as you tested, know that it's not too late to consider writing as a helpful hindsight strategy. Once a defect appears that you can't replicate, write the steps you believe you took. When you see the steps on paper, the options you selected during testing, along with other test factors, may return to your memory.

There was one instance where I encountered a nasty system crash, but I hadn't recorded my steps while testing. So I sat back and wrote the steps I thought I had taken. There were three, but I wasn't clear in which order I had executed them. By concentrating on the three steps, I remembered one step I'd forgotten (I had changed the paper size before I queued the print job). It turned out that if I changed the paper size on the print dialog, our application hit an error. I felt I should have remembered such an important element, but sometimes we don't notice all the steps we take. That's why writing as we test can help.

Solicit Help

Find out who coded the area and *talk* to them. Tell them what's happened and see if they have ideas about what may have gone wrong. I've talked to developers about suspicions of a bug and found that they too have encountered "something" along the way. By brainstorming together, we usually solve the mystery.

But remember that there are several important dynamics going on between quality assurance and development. Every conversation you have with a developer establishes, reinforces, or detracts from your credibility. While it's only human to admit you can't recall every step you took—particularly if your testing session was long—repeating this type of seemingly haphazard approach too often may cause you to lose credibility. Therefore, state only what you know and be factual. Leave your guesses out of the conversation. Remember: You're now seeking *their* guesses as to what may have gone wrong.

Once while testing a Web site, I encountered what seemed to be intermittent, unacceptably slow performance. I investigated and ruled out server-specific issues, transaction-specific issues, and other inklings. I began letting a developer know each time I encountered the slow response. We were convinced there was an issue and became united in finding our ghost. Eventually, we discovered that cached product data was being reloaded during the times of poor performance. The reload generated a flood of requests to the database. Problem found. Case closed.

Keep It Private

While soliciting help can be useful, there are also reasons you might not want to ask for help. Reporting another elusive defect might well add to overall team stress at a bad time, such as just before a release. The relevant developer might be poor at this type of debugging or unwilling to assist. If the two of you end up unable to replicate the bug, two people's time has been lost. Consider all the angles before asking for help.

But you must always ask yourself about likelihood and risk. How likely is this bug to occur in production? Would you be negligent in not mentioning this bug? How harsh is the bug for your users? If the answer to any of these questions is "very," you should shamelessly pursue the replicating of the bug by soliciting whatever help you can.

When You Can't Leave Well Enough Alone

Ghosts leave trails. Even when you're unable to replicate one of your ghosts, inklings of one bug should generate ideas about where you should look for others, i.e., where you should focus your testing. Spend time with your inklings, for each time you do, you demonstrate confidence in your own gut about where there are issues. And you build strength in your ability to replicate the truly nasty beasties.

I've kept many of my old ghosts in the closet, well away from the eyes of the world. But at times, when I've had a spare moment or a quiet lunch at my desk, I freely admit I've tried to conjure them up again. STQE

Karen Johnson is a business systems consultant for Baxter Healthcare Corporation. She has more than sixteen years of experience in computer software, including client-server and Web testing activities such as installation, multiple-user testing, and data replication. She can be reached at knj_karen@yahoo.com.