QUICK LOOK

- 4 tips to help your bug reports get noticed
- How to provide key players with information they can *really* use

Writing Effective Bug Reports

by Elisabeth Hendrickson

Have you ever had a bug returned to you for more information? Have you ever found a critical bug only to have it deferred to another release?

Every bug report you file is a written communication to the project team about the quality of the software you're testing. Often, your ability to communicate about an application's bugs—not the inherent severity of the bugs themselves—determines whether or not the bugs get fixed.

If this is a scary thought, you may be thinking, "But wait! I hate writing, and I'm no good at it. How can the fate of the bug be determined by the writing of the bug report?" It may be tempting to believe that bugs speak for themselves—that any right-thinking person will automatically see that a given bug is horrid enough that it should be fixed. Unfortunately, that's not the case.

But here's the good news: your ability to communicate effectively with the software developers and the project team is *not* predetermined by how well you did in high school English class.

It isn't about writing flowing prose with interesting words. It isn't even about good grammar and spelling. It's about speaking plainly, taking just enough words to make your point. Too many words, and your point is lost amid the ramble. Too few words, and others fill in the gaps with their own assumptions—often to the detriment of the software. It's also about knowing exactly what it is that you need to communicate. If you aren't sure exactly what the bug is, then no matter how well you write the bug report, no one else will know what the bug is either.

This *Bug Report* discusses four things you can start doing now to improve the chances that people will listen to your bugs.

1: Know Your Audience(s)

Any writing class will tell you that you must know who you're writing for. Bug reports are no exception.

There are at least two audiences for every bug report: the person who must fix the bug and the person or group that decides the fate of the bug. While one person is sometimes going to be responsible for both jobs, there are still two different audiences; they just happen to be in one body.

Your first audience—the person who must fix the bug—needs clear, unambiguous steps to follow in order to reproduce the symptoms. The more information, the better. For the purposes of this bug report, we'll call this person the "Developer." The Developer needs precise detail about what you did and what you saw.

Your second audience—the person or group who decides on the fate of the bug—needs to understand the consequences of choosing not to fix the bug. This audience needs a pithy statement to grab their attention and a discussion of the implications of the bug. For the purposes of this Bug Report, we'll call this audience the "Bug Review Committee." Your role in getting bugs fixed is to help the Bug Review Committee see that the risks of not fixing the bug far outweigh the possible risks of fixing it.

The more you understand about how your Developers and Bug Review Committee work, the more you can tailor your bug reports to their needs. Make an effort to get to know your audiences personally. If you can attend Bug Review Committee meetings, do so. You'll learn a lot about how your reports' audiences think.

2: Choose a Good Title

The short phrase that describes the bug is usually called the *bug title* or *bug description*. This is the most important part of the bug report. Members of the Bug Review Committee often decide a bug can be deferred based on the title alone; if the title is weak, committee members may determine that it's not worth spending any more time on the bug. (After all, there are 145 more to review in the next two hours.)

Here are some examples:

Good: *Crash on exit after timeout*

Too Long: *Program crashes when you choose exit from the file menu just after the database has become unavailable and you were saving changes to a record*

Not Enough Detail: *Program crashes*

Too Vague: *Problem when database offline*

The title also becomes a way for the project team to refer to and search for the bug. People tend to remember words better than numbers. So rather than remembering bug 23423, people will remember the bug that "Can't Install on Windows 2000" and will use those keywords to search for the bug later.

It's difficult to write a good, concise bug title. Expect to spend more time crafting the perfect bug title than on writing the rest of the bug report. Make sure that the title is short enough to display entirely on the bug screen (without scrolling) and in reports generated by the bug tracking system. The title doesn't have to be

grammatically perfect—it needs to be short and to the point.

3: Write Clear, Unambiguous Steps

The *steps* you give the Developer will provide information about where the bug lives, so it can be found and fixed. They also provide information to the Bug Review Committee about the circumstances under which the bug occurs.

Just right:

1. Launch the client
2. Call up a record
3. Change the record but don't save your changes yet
4. Take the database server offline
5. Attempt to save the record
6. Receive a timeout error
7. Exit the client

Result: Crash

Too imprecise (too much room for misinterpretation):

Take the database offline, save, then exit. Crash.

Too much extraneous information (can't tell what's significant to the bug):

1. Launch the client
2. Query the database for new entries
3. Launch a browser
4. Read the news on yahoo.com
5. Quit the browser
6. Choose an entry
7. Change the category from "vegetable" to "fruit"
8. Take the database offline
9. Attempt to save the record
10. Receive a timeout error
11. Exit the client

Result: Crash

In this example, the tester transcribed everything he did before finding the bug. But he didn't check to see if every step, like getting news from yahoo.com, was necessary.

If you work to get the number of steps down to those that are absolutely required, developers are much less likely to tell you that they can't reproduce your bug, and Bug Review Committees are much less likely to decide that "no one would ever do all that!"

But what if every step *was* necessary? If the bug only manifests after you perform apparently irrelevant steps, call out those steps in the bug report. You could write "necessary step" after the apparently illogical steps, or you could add a note at the beginning of the bug report: "NOTE—Every step here is necessary to reproduce the bug."

Writing clear steps also helps when it's time to verify the fix, especially if another tester must do the verification.

4: Explain the Effect of the Bug, Not Just the Symptoms

Some bug reports are misleading. On the surface the bug appears innocuous, but if you examine the implications of the bug you discover a very serious problem. If you were on the Bug Review Committee, which bug would you advocate fixing first?

1. A report that "an annoying dialog prevents you from closing application"

2. A report that "the application hangs on exit"

These are the same bug. The difference is entirely in how the tester wrote the bug report.

The "annoying dialog" in this case was the window that appears in Windows when it cannot exit a process ("This Windows application cannot respond to the End Task request..."). The tester found the problem by attempting to shut down the machine without first exiting the application. The application was not waiting for input from the user, so there was no reason for it to fail to exit. In fact, this symptom indicated deeper problems—problems that were almost missed when the first bug report about an "annoying dialog" was deferred.

There were two problems with the "annoying" bug report. First, it was imprecise. Had the tester included the text of the "annoying dialog" in the steps, the decision makers would have recognized the dialog as a serious problem rather than a minor annoyance. Second, the report didn't indicate the implication of the bug: the application was hanging.

Conclusion

We all want our work to make a difference. We want to know that the final release of the software was better because we worked on it. Our ability to communicate about the bugs we find is likely to be the determining factor in whether or not we have as much of an impact on the final version of the software as we'd hoped.

So when you write bug reports, remember your audience, choose a good title, document the steps clearly, and explain the implications of the bug. Your bug reports will be better for the extra effort you put into them, and more of the bugs you report will get fixed. And ultimately that's the point—to get more bugs fixed before they can hurt the users. STQE

Elisabeth Hendrickson (esh@qualitytree.com) *is an independent consultant specializing in software quality assurance and management, with twelve years of experience working with leading software companies. An award-winning author, Elisabeth is a frequent speaker at major conferences on software quality and management. You can read more about her ideas on quality and testing at* www.qualitytree.com.

STQE magazine is produced by STQE Publishing, a division of Software Quality Engineering.