

# Tips on Writing Effective Bug Reports

How often do we see the developers requiring more information on the bug reports filed by us? How often do we need to spend more time investigating on the issue after the bug report has been filed? How often do we get to hear from the developers that the bug is not reproducible at their end and we need to improvise on the *Steps to Reproduce*?

Conclusively, we end up spending more time on these issues rather than investing more time testing the system. The problem lies in the quality of bug reports. Here are some areas which can be improved upon to achieve *that* perfect bug report.

## **The Purpose of a Bug Report**

When we uncover a defect, we need to inform the developers about it. Bug report is a medium of such a communication. The primary aim of a bug report is to let the developers see the failure with their own eyes. If you can't be with them to prove application failure, make available detailed instructions so that they can make it fail for themselves. The bug report is a document that explains the gap between the expected result and the actual result and the details on how to reproduce the scenario.

## **After Finding the Defect**

Draft the bug report immediately after finding the bug. Do not leave it for the end of test or end of day. It might be possible that you might miss out on some point. Worse, you might miss the bug itself.

Invest some time to diagnose the defect you are reporting. Think of the possible causes. You might land up uncovering some more defects. Mention your *discoveries* in your bug report. The programmers will only be happy seeing that you have made their job easier.

Take some time off before reading your bug report. You might feel like re-writing it.

## **Bug Report Summary**

The summary of the bug report is the reader's first interaction with your bug report. The fate of your bug heavily depends on the attraction grabbed by the summary of your bug report. The thumb-rule is that every bug should have a one-liner summary. It might sound like writing a good attention-grabbing advertisement campaign. But then, there are no exceptions. A good summary will not be more than 50-60 characters. Also, a good summary should not carry any subjective representations of the defect.

## **Bug Report Language**

Do not exaggerate the defect through the bug report. However, do not downplay it either. However nasty the bug might be, do not forget that it's the bug that's nasty, not the programmer. Never offend the efforts of the programmer. Use euphemisms. "Dirty UI" can be made milder as "Improper UI". This will take care that the programmer's efforts are respected.

Keep It Simple & Straight by using simple language.

Keep your target audience in mind while writing the bug report. They might be the developers, fellow testers, managers, or in some cases, even the customers. The bug reports should be understandable by all of them.

### **Steps to Reproduce**

The flow of the Steps to Reproduce should be logical:

1. Clearly list down the pre-requisites.
2. Write generic steps. For example, if a step requires the user to create file and name it, do not ask the user to name it "XYZ file". It can be better named as "Test File".
3. The Steps to Reproduce should be detailed. For example, if you want the user to save a document from Microsoft Word, you can ask the user to go to File Menu and click on the Save menu entry. You can also just say "save the document". But remember, not everyone will not know how to save a document from Microsoft Word. So it is better to stick to the first method.
4. Test your Steps to Reproduce on a fresh system. You might find some steps that are missing, or are extraneous.

### **Test Data**

Strive to write generic bug reports. The developers might not have access to your test data. If the bug is specific to a certain test data, attach it with your bug report.

### **Screen Prints**

Screen prints are quite an essential part of the bug report. A picture makes up for a thousand words. But do not make it a habit to unnecessarily attach screen shots with every bug report. Ideally, your bug reports should be effective enough to enable the developers to reproduce the problem. Screen shots should be a medium for verification.

If you attach screen shots to your bug reports, ensure that they are not too heavy in terms of size. Use a format like jpg or gif, but definitely not bmp.

Use annotations on screen shots to pin-point at the problems. This will help the developers to locate the problem at a single glance.

### **Severity / Priority**

The impact of the defect should be thoroughly analyzed before setting the severity of the bug

### ***Dashboard October 2006 Issue***

report. If you think that your bug should be fixed with a high priority, justify it in the bug report. This justification should go in the *Description* section of the bug report. If the bug is the result of regression from the previous builds/versions, raise the alarm. The severity of such a bug may be low but the priority should be typically high.

## **Logs**

Make it a point to attach logs or excerpts from the logs. This will help the developers to analyze and debug the system easily. Most of the time, if logs are not attached and the issue is not reproducible on the developer's end, they might revert back asking for logs.

If the logs are not too large, say about 20-25 lines, you may paste it in bug report. But if it is large enough, add it to your bug report as an attachment, else your bug report will look like a log.

## **Related tips:**

- If your bug is randomly reproducible, just mention it in your bug report. But don't forget to file it. You can always add the exact steps to reproduce anytime later you (or anyone else) discover them. This will also come to your rescue when someone else reports this issue, especially if it's a serious one.
- Mention the error messages in the bug report, especially if they are numbered. For example, error messages from the database.
- Mention the version numbers and build numbers in the bug reports.
- Mention the platforms on which the issue is reproducible. Precisely mention the platforms on which the issue is not reproducible. Also understand that there is difference between the issue being not reproducible on a particular platform and it not being tested on that platform. This might lead to confusion.
- If you come across several problems having the same cause, write a single bug report. The fix of the problem will be only one. Similarly, if you come across similar problems at different locations requiring the same *kind* of fix but at different places, write separate bug reports for each of the problems. One bug report for only one fix.
- If the test environment on which the bug is reproducible is accessible to the developers, mention the details of accessing this setup. This will help them save time to setting up the environment to reproduce your bug.
- Under no circumstances should you hold on to any information regarding the bug. Unnecessary iterations of the bug report between the developer and the tester before being fixed is just a waste of time due to ineffective bug reporting.