When Testers Should Consider a Bug a Duplicate

By **Michael Stahl** - January 15, 2018
https://www.stickyminds.com/article/when-testers-should-consider-bug-duplicate

Summary:
When can a bug report be considered redundant because it is already reported in the bug management system? If you ask the developers, if two bugs are caused by the same mistake in the code, it's enough to report one of them. But Michael Stahl has good arguments from a tester's perspective about why it's better to err on the side of over-reporting bugs.

I once asked an acquaintance, a math professor, what he did to get his PhD.

"They gave me a problem to solve," he said.

"And?" I asked.

His answer: "It is not a problem anymore."

In software testing, there are a good number of topics that can be considered solved when there is a well-known and commonly accepted way to deal with them. One such example is bug management.

In theory, it's a clear-cut process: There is a triage meeting attended by development, the validation project manager, and other interested parties. Bug reports are checked to include a clear description, logs, screenshots as needed, and a reasonable description of how to reproduce the bug. Next, there is a discussion about the bug severity. Sometimes it's an easy call, and sometimes it's complicated and requires deep understanding of the underlying technology. It's common to have arguments—possibly heated ones—about the severity of a certain bug. As the release date approaches, a fair amount of politics play a role as well. Once the severity is agreed upon, the priority is set: Do we fix the bug right away, or can it wait? Here, too, there is an opportunity for arguments. But generally speaking, the process is clear and mostly works.

Consequently, I am repeatedly surprised by how specific aspects of the process never seem to have been fully agreed on and are still a contentious point, even though they should have moved to the "not a problem anymore" category a long time ago.

One such aspect is the question of duplicate bugs. Specifically, when can a bug report be considered redundant because it is already reported in the bug management system? In such a case, the new report is superfluous, does not merit any discussion, and can be closed right away.

A bug marked as a duplicate is perceived as a black eye to validation (and it should be seen this way!). It means the validation team is unprofessional: The tester should check in the bug reporting system whether the bug is already reported and refrain from adding "noise" to the data by repeating the reporting of the same bug. That's why validation opposes marking a bug as duplicate and looks for justification for the new bug. Development, on the other hand, wants to minimize the number of open

bugs, so closing a bug without having to work on it is a good thing. Disagreement flourishes, and the bug triage meeting turns into an ineffective quarrel.

When *should* we consider a bug a duplicate?

It depends, of course, on whom you ask. If you ask the developers, if two bugs are caused by the same mistake in the code, it's enough to report them once. The second bug will be fixed when the first bug is fixed. As the developers see it (and many times the project manager thinks so, as well), the claim that in this case both bugs are valid distorts the information we have about the quality of the code.

Here is a convincing example (if you are a developer): Assume you have an application for storing and reporting students' grades. The code for extracting grades from the database uses the student's Social Security number (SSN) to identify the student. The application allows the student to identify themselves using a variety of methods in addition to the SSN: full name, security PIN, cellphone number, and credit card number. In the first stage, the application uses the identifying information to extract the SSN from the student's personal record. Then the SSN is used to extract the grades. (All you database experts out there who are actively pulling your hair out, crying, "This is *not* how it's done!" Chill! I said *assume*.)

Five test cases were defined at the system test level. Each test case checks the ability to extract grades using a different identification method. Now, what happens if there is regression in the code that extracts the grades from the second table? All five test cases will fail!

What's the quality level of the code in this case? If we report five bugs, we pass a message that the grade-extract module is in severe regression, and it may have a serious impact on the project. But this is obviously incorrect. There is only one error in the code, and the whole thing will probably be solved in half a day, including pre-check-in tests.

So there is a good argument why four out of the five bugs are duplicates. But here is the counter argument.

How confident can we be that the reason for all five failures is the same one? Indeed, in the example given here, it makes a lot of sense. But between us testers: If they messed up the grades extract code, maybe they also messed up the SSN extraction. I have full confidence in the developers. I am sure they know how to create two bugs in one release!

If we close four of the bugs as duplicate, it may be that only one error will be fixed. The other error will have to wait to be found until we run the tests again. Because no bug was opened against those test cases, we won't be doing confirmation tests. If the grade-extraction tests are low-priority test cases, it may take quite a while until we run them again. For all we know, we may never run them again before the product is released (at which point, Murphy says, the bug will appear immediately).

My position is that testers need to report what happened. Why it happened—the root cause—is irrelevant when we discuss bug reporting. The developer's claim of duplicity relies on the assumption that what happened at the system level and why it happened at the code level are the same for all bugs. The counterclaim of the testers is that what we did at the system level is different.

I maintain that a bug should be considered a duplicate if and only if both the action done and the erroneous result are the same.

Here's another tell-tale sign when a bug is not a duplicate. Usually, a duplicate bug is reported when two testers identified the same problem and both reported it without first checking if it's already in the system. It is therefore reasonable to say, with high level of confidence, that when both bugs were open by the same tester, it is not a duplicate. Why would the same tester report the same issue twice? It just adds extra work for the tester, who for sure remembers the first report. Clearly, the manifestation of the bug, either what we did or what happened, is not exactly the same—even if the developers think it is.

It's important to have a discussion about this topic, detached from a specific case, and arrive at an agreement with the involved parties about when a bug should be considered a duplicate. First, it will avoid repeating discussions in bug triage, but more than this, it will prevent negative phenomena that happen when there is a strong tendency to mark bugs as duplicates. I experienced two such phenomena:

- Testers decide not to report some incidents, because "it will be closed as a duplicate anyway"
- Testers take this attitude further: When they encounter a bug and think they understand its overall influence on the code behavior, they don't open bugs on errors they conclude are a result of the first issue, even when the new bugs' manifestation is totally different than in the first bug

It's better to be skeptic and picky in tagging bugs as duplicates. The damage caused by a few duplicate bugs is lower than the damage caused by unreported bugs. It is better to err on the over-reporting side.

Whenever a bug is encountered, if it is not *exactly* the same as a previously reported bug, create a new report. If indeed both are caused by one error, the developers got one for the price of two. What is important to do in such a case, though, is to link the bugs in the bug-reporting system (most systems support such linking). This will tell the developers you think these two bugs may be related or are possibly a different manifestation of the same error. But links are not enough—developers tend to miss them. Write clearly in the text description of both bugs, "It is possible that this bug is caused by the same root cause of bug X, linked to this bug."

All that is now left for me to do is to hope that this topic too will move to the "not a problem anymore" category.