How to Respond to Retest Requests without a Clear Bug Fix

By **Michael Stahl** - April 27, 2020
Share URL
Bookmark

Summary:
After finding and reporting a bug, a tester may get this response from a developer: "Please rerun the test on the latest version of the code and check if the bug still reproduces." This seems like a rational request; just as a change can cause a bug to appear, it can also fix a bug. But is following up the responsibility of the tester or the developer? And if the bug is no longer there, how do you classify and close it?

No one likes to do redundant work. But sometimes rework is a good idea—or even necessary.

For example, recently I was in the kitchen, planning to engage in my hobby of baking bread. I checked the pantry and found no flour. I asked my wife, “Honey, are we out of flour?” “Check again,” she offered.

Check again? What is there to check? I just did, didn’t I? What could have changed? Did a fairy arrive and place some flour in the pantry? But to keep the peace, I checked again, scanning all the shelves.

“No flour!” I reported. At this stage my wife sighed, got up, went to the pantry … and showed me two sacks of flour on the shelf right at eye level.

Okay, maybe this is a bad example. Here is a better one.

We found a bug and reported it. A week or two go by, and nothing seems to happen. When we asked what’s up, we got this: “Please rerun the test on the latest version and check if it still reproduces.”

What’s to run again? We saw the failure already! What could be different? And then we got the ultimate answer: “We changed a lot of things in the code.”

On the surface, this seems like a rational answer. Just as a change can cause a bug to appear, it can also fix a bug. But somehow, deep inside, it feels like someone is playing for time in order to explain why the bug is not addressed, or that there is some unsubstantiated hope that a miracle happened and because the developers touched a few lines of code in a file related to our bug, the bug would go away.

To some degree, it also feels as if the message is that the developers’ time is more precious than the testers’.

I wondered what other testers think about this issue and how they deal with these situations. I raised this as a question on **a web forum** and received many responses.

I must admit that I was expecting a lot of support, with other testers sharing the frustration and adding cynical wisecracks about the unacceptable and conceited behavior of developers. It could have been real fun. Instead, I got something better: food for thought and a new outlook on the topic.

First, there is a general agreement the developers should give a viable explanation for the bug's possible disappearance. A vague answer of "We changed a lot of things" is not as easy to accept as something more useful and specific, like, "We did a major refactoring of the code in this module. In fact, more than half the code was rewritten. We defined new classes and replaced the main algorithm. It is very likely that the failing line of code is not in the code anymore." I would even accept significantly simpler changes to the code as a logical explanation of why the bug is possibly gone, if it makes sense. Something like, "We improved the input checking in some of the APIs" is a perfectly good explanation.

Fine. I am convinced that the bug may have disappeared. Why don't you test it yourself and tell me that the bug is gone? Why do *I* have to do it?

It turns out that sometimes this is a reasonable request. There are cases where it is much easier for the testers to run a test than for the developers. It may be because the test environment is already set up and all the tester needs to do is to invoke an automated test. It may be that the bug was found on a hardware version that the developers don't have, such as a legacy version. Sometimes it's a long and detailed manual test where the developer, who is not experienced in running it, will lose a lot of time getting the test to run correctly and has a good chance of making mistakes that will invalidate the results or miss triggering the bug.

When developing a new feature, the code goes through rapid changes between released versions, so indeed there is a chance that the bug will disappear. In this case, we can look at the developer's request not as "Show me that the bug still exists," but more like "We believe we fixed it and ask you to run a confirmation test."

If we really want to keep a positive attitude about it, we can even see the request for rerunning a test as a compliment—the developers don't trust themselves to run the test correctly. Yes, I know it sounds a bit like when your dad told you, "Please mow the lawn. You do it so much better than anyone else!" But sometimes these requests are indeed out of real appreciation of the tester's capabilities.

Alternately, we can be proactive and design the test automation platform so that developers can use it easily to run tests. If this is achieved, they won't need to come to us to check if a bug is no longer there. Forum commenter **Michael Durrant** phrased it well: "Respond to requests to test by enabling the requestor to run the tests." Of course, this will work only for bugs that were found by the automated tests; it does not solve the problem of manual tests or exploratory tests. But it is likely to reduce the number of cases where testers need to be the ones to do the test rerun.

Generally, the answers in the forum, especially those that came from developers (which is an interesting thing by itself—developers who spend time in a forum for testers) tended toward giving a logical explanation for the request and emphasizing that it is not done out of disrespect or laziness.

User **Kate Paulk** wrote that there's no ill will on the dev team's side; it's often a case of there being a better use of resources. And when I **posted the forum discussion to Facebook**, someone else suggested, "Some bugs are generated by the integration of different parts of the code, and it’s legitimate for the developer to ask for test rerun, without a specific fix, after code changes were made in that area.”

In truth, if this is the case, we should not be satisfied in running just the specific test that found the bug. If the bug disappeared as a side effect, maybe there were other side effects, some of which are not that positive. When a bug disappears, it’s a good hint that we need to test more in that area—sometimes just a bit, and sometimes more, depending on the level of change that was done.

Some of the participants in the discussion noted that if this type of request is frequent, it’s an indication of some basic issue in the organizational culture or in tester-developer communication, and teams will do well to discuss and try to understand the high number of retest requests. Are the bug reports not clear enough or missing details? Did the developers try to reproduce the bug at all? Did they contact the person who reported the bug? Do they understand the cost of the retest request in terms of the time it takes and the context-switch impact? A discussion will help each side get a better perspective about the constraints and limitations of the other team, will usually increase the mutual respect, and will improve cooperation.

It may be worthwhile to track how many times the developers are right in their request—that is, that the bug is indeed gone. If in most cases they are right, it indicates they ask for a retest only when they have a pretty good reason to assume the bug was fixed indirectly. They should have documented this—and that’s something they may want to work on—but it’s more as if they asked for a confirmation test after a bug fix, so it should not be seen as a baseless retest request. If in most cases the bug does reproduce, then developers may be hoping for miracles, and this is a totally different management problem. Another good reason to track the number of fixes-as-a-side-effect is that a high rate of occurrence points to highly coupled code, and this is something that should be noted and improved.

While I was at it, I also requested feedback on how to close such a disappearing bug in the bug management system. Project managers tend to classify these bugs as “not reproducible,” but this is incorrect! The bug is highly reproducible on the version of the software where we found it! It does not reproduce *on a different version*.

Since we **track the number of irreproducible bugs in order to improve**, counting a miraculously resolved bug as a testing issue is unfair and incorrect. However, in most bug management systems, trying to close the bug as “fixed” will bring up a whole set of questions that can’t be answered in the case of a spontaneous fix: what file was changed, what was the root cause, what exactly was the fix, etc.

One option I saw is to mark such a bug as “rejected,” and in a companion field to explain the reason, put “No longer seen” or “Can’t reproduce on new version.” The capabilities and options offered by your bug management system will define how you close such a bug. Any option is fine, as long as it clearly marks the bug for what it was and will not mark it as an indication of failure by the testers.