Exit Criteria, Software Quality, and Gut Feelings

Michael Stahl (michael.stahl@intel.com)
https://www.stickyminds.com/article/exit-criteria-software-quality-and-gut-feelings

A few months ago I happened to meet an old friend who is also a software tester at a social gathering. Soon enough we started to talk shop, and my friend shared his bleak situation.

“Tomorrow we have a project meeting to approve the release of our latest version,” he said. “I am going to recommend we don’t release it, but I know I will be ignored.”

“Why this recommendation?”

“Because I have a strong gut feeling that the last version we got—the one about to be released to our customers—is completely messed up. But the product manager just follows our exit criteria list, which we meet, and ‘gut feeling’ is not considered to be solid data.”

I inquired about the source of this gut feeling and got the whole story: Two weeks ago his team received a build for a last test cycle before the official release. Theoretically, everything was supposed to be perfect. But on the first day of testing, the testers found two critical bugs.

The development team rushed to fix them and released a new build. However, for two weeks, the cycle repeated: The testers would find one or two critical bugs and the developers would release a new build with fixes. This created the problem my friend had.

“By our exit criteria, we can’t release a version that has critical bugs,” he explained. “But the development team fixed every bug we found, so at least for this moment in time, there are no open critical bugs. In the last two weeks I got eight new builds. Do you think we got even close to testing the product properly? Based on the quality level of builds in the last two weeks, I can guarantee that the build we got today contains a good number of additional critical bugs. If I get a few more days to test it, we will find them, but release day is tomorrow, and on paper everything is fine.”

What can we do to deal with such situations? How do we translate our professional intuition, our gut feeling, into hard data?

Creating Comprehensive Exit Criteria

The ideal way to communicate these feelings is to be proactive. You need exit criteria that are a bit more sophisticated than a simple bug count and bug severity. One has to add trends: the rate of incoming new bugs, the rate of bugs being closed, and the predicted bug count based on the number of yet-to-be-executed tests.

Here are some examples for these criteria:

- The number of critical bugs opened in the last test cycle is less than *w*
- The count of new bugs found in the last few weeks is trending down at a rate of *x* bugs/week
- Testing was executed for at least y days on the last version without any new critical bugs and not more than z new high-severity bugs

Note, however, that bug counts and trends are not covering all the quality aspects of a product. An exit criteria list that tracks only bug statistics does not guarantee much; you can meet all the bug-related criteria and still have a bad product. Your requirements, product manuals, support readiness, hardware quality, etc., may not be as good as they should be. Additionally, the limits you put in the exit criteria (e.g., maximum open bugs) may be too relaxed; it takes some experience to know how to set them for a specific product line.

A good exit criteria list provides an orderly list of attributes that research and experience showed to have impact on product quality. Improving the results on these attributes is known to have a positive impact on product quality.

For example, I found this outline for release criteria useful. You can also download my exit criteria spreadsheet. (Make sure to read the "How to use this file" text before adopting it wholesale.)

Monitoring Quality and Forecasting Status

A list of exit criteria can be used as a simple go/no-go decision tool when a release date arrives. However, it can be used much more effectively if it is referred to all along the product development timeline.

For example, let's say you have a criterion of "No more than one hundred open bugs with medium or low severity." Assume you also have the following information (you can get some of it from the bug database, and other pieces come from the development and test teams):

- Current open bugs: 300
- Time to release: 10 weeks
- Projected new bug submissions in the next 10 weeks: 100
- Number of engineers assigned to fix bugs: 2
- Average time to fix a bug: 0.5 days (so in the next 10 weeks, two engineers can fix 200 bugs)
-

From these data you can estimate that the bug count expected on the release date is two hundred. Your exit criteria, to remind you, is no more than one hundred.

This means that by the current estimation, which is done ten weeks ahead of release time (!), you can proactively raise a red flag and inform the development manager that they are at risk of not meeting a specific exit criterion. They need to assign more people to work on bug fixes.

While the example can be seen as fabricated (well … it is), it does convey an important message: Exit criteria are a useful tool to monitor the product quality at any given time and to forecast the expected status at release date. When used properly and consulted throughout the development time, these criteria can help avoid at least some of the firefighting we all experience just before a release date.

Many of the criteria in the list I propose are simple to track and do not rely heavily on estimations as the given example. If your exit criteria require that all the committed design documents are published, reviewed, and updated, the simple act of collecting the actual data will give an early warning when too many documents are not completed.

Getting Your Team On Board with Exit Criteria

You can't just set the exit criteria or their limits without wide agreement. Any team that will need to deliver in order to meet the criteria must agree that the criteria are worthwhile and the limits are fair and achievable.

You also must set the criteria as early as possible in the project timeline. At that time all the stakeholders are optimistic, full of good intentions about quality, and not under pressure. They are in a mindset that allows discussing the criteria and limits in an objective manner. If you try to agree on criteria and limits a few weeks before release time, the limits that the development team will agree to will be significantly influenced by the current state of affairs rather than what good quality is.

Back to my friend.

Adding exit criteria ahead of time would have revealed a clear stop sign to the project managers, and my friend could have enjoyed the party. But what can be done in a case similar to the one described here, where such criteria were not established in advance?

I suggested to my friend that he "translate" his gut feeling to hard data and present it in a clear manner: "This is the current status: There are too many new critical bugs in the code and not enough time for proper testing. Based on these data, I conclude that the version quality is not fit for release." These aren't feelings or intuitions, but a professional opinion based on facts, experience, and in-depth knowledge of development processes.

I called a few weeks later to hear how it went. My friend had presented graphs:

- The number of new critical bugs on a daily basis (one or two per day; no improvement trend)
- The time the testers had to test each new release in the last two weeks (one or two days)
- The percent of tests that were executed on the last version (about 25 percent of the planned tests)

This was enough to give everyone a bad gut feeling, and the release was delayed.

The moral of the story? It's possible to translate gut feelings about quality to hard data. However, as a rule you want to avoid such situation altogether. The early definition of exit criteria, combined with data, lets you monitor quality throughout the project's lifetime and avoid potential last-minute disasters.