# How to gauge the quality of a software product

Asked 16 years, 4 months ago Modified 11 years, 1 month ago

Viewed 2k times

10

I have a product, X, which we deliver to a client, C every month, including bugfixes, enhancements, new development etc.) Each month, I am asked to err "guarantee" the quality of the product.

For this we use a number of statistics garnered from the tests that we do, such as:

- reopen rate (number of bugs reopened/number of corrected bugs tested)
- new bug rate (number of new, including regressions, bugs found during testing/number of corrected bugs tested)
- for each new enhancement, the new bug rate (the number of bugs found for this enhancement/number of mandays)

and various other figures.

It is impossible, for reasons we shan't go into, to test everything every time.

So, my question is:

How do I estimate the number and type of bugs that remain in my software? What testing strategies do I have to follow to make sure that the product is good?

I know this is a bit of an open question, but hey, I also know that there are no simple solutions.

Thanks.

testing

Share

Improve this question

Follow

edited Nov 14, 2013 at 22:01



user1228

asked Aug 18, 2008 at 20:18

Matthew Farwell

**61.7k** ● 18 ● 132 ● 173

## 7 Answers

Sorted by: Highest score (default)


I don't think you can ever really estimate the number of bugs in your app. Unless you use a language and process that allows formal proofs, you can never really be sure. Your time is probably better spent setting up processes to minimize bugs than trying to estimate how many you have.

One of the most important things you can do is have a good QA team and good work item tracking. You may not be able to do full regression testing every time, but if you have a list of the changes you've made to the app since the last release, then your QA people (or person) can focus their testing on the parts of the app that are expected to be affected.

Another thing that would be helpful is unit tests. The more of your codebase you have covered the more confident you can be that changes in one area didn't inadvertently affected another area. I've found this quite useful, as sometimes I'll change something and forget that it would affect another part of the app, and the unit tests showed the problem right away. Passed unit tests won't guarantee that you haven't broken anything, but they can help increase confidence that changes you make are working.

Also, this is a bit redundant and obvious, but make sure you have good bug tracking software. :)

Share Improve this answer

Follow

answered Aug 18, 2008 at 20:43

Herms

38.7k ● 13 ● 79 ● 104

> Bebugging is the answer to getting a statistically plausible estimate of latent bugs , en.wikipedia.org/wiki/Bebugging – Tim Williscroft Apr 14, 2010 at 3:54

2

The question is who requires you to provide the stats.

If it's non-technical people, fake the stats. By "fake", I mean "provide any inevitably meaningless, but real numbers" of the kind you mentioned.

If it's technical people without a CS background, they ought to be told about the halting problem, which is undecidable and is simpler than counting and classifying the remaining bugs.

There's a lot of metrics and tools regarding software quality (code coverage, cyclomatic complexity, coding guidelines and tools enforcing them, etc.). In practice, what works is automating as much tests as possible, having human testers do as many tests that weren't automated as possible, and then pray.

Share Improve this answer

Follow

answered Aug 18, 2008 at 20:36

Yossi Kreinin

246 ● 1 ● 3

I think keeping it simple is the best way to go. Categorize your bugs by severity, and address them in order of decreasing severity.

This way you can hand over the highest-quality build possible (the number of significant bugs remaining is how I would gauge the quality of the product, as opposed to some complex statistics).

Share Improve this answer

Follow

answered Aug 18, 2008 at 20:27


conmulligan

**7,138** ● 6 ● 34 ● 45

---

Most of the agile methodologies address this dilemma pretty clearly. You can't test everything. Neither can you test it infinite number of times before you release. So the procedure is to rely on the risk and likelihood of the bug. Both risk and likelihood are numerical values. The product of both gives you a RPN number. If the number is less than 15 you ship a beta. If you can bring it down to less than 10 you ship the product and push the bug to be fixed in a future releasee.

How to calculate risk ?

If its a crash then its a 5 If its a crash but you can provide a work around then its a number less than 5. If the bug reduces the functionality then its a 4

How to calculate likelihood ?

can you re-produce it every time you run, its a 5. If the work around provided still causes it to crash then less than 5

Well, I am curious to know whether anyone else using this scheme and eager to know their milage on this.

Share Improve this answer

Follow

answered Aug 18, 2008 at 20:44

rptony

**1,024** ● 2 ● 12 ● 22

---

0

How long is a piece of string? Ultimately what makes a quality product? Bugs gives some indication yes, but many other factors are involved, Unit Test coverage is a key factor in IMO. But in my experience the main factor that effects whether a product can be deemed quality or not, is good understanding of the problem that is being solved. Often what happens is, the 'problem' that the product is meant to solve is not understood correctly and developers end up inventing the solution to a problem they have flesh out in their head, and not the real problem, thus 'bugs' are made. I am a strong proponent of iterative Agile development, that way the product is constantly access against the 'problem' and the product does not stray to far from its goal.

Share Improve this answer

Follow

answered Aug 18, 2008 at 20:41

Dan

**29.4k** ● 44 ● 151 ● 209

0

The questions I heard wer, how do I estimate the bugs in my software? and what techniques do I use to ensure the quality is good?

Rather than go through a full course, here are a couple approaches.

**How do I estimate the bugs in my software?**

Start with the history, you know how many you found during testing (hopefully) and you know how many were found after the fact. You can use that to estimate how efficient you are at finding bugs (DDR - Defect Detection Rate is one name for this). If you can show that for some consistent time period, your DDR is consistent (or improving) you can provide some insight into the quality of the release by guessing at the number of post-release defects that will be found once the product is released.

**What techniques do I use to ensure the quality is good?**

Root cause analysis on your bugs will point you to specific components that are buggy, specific developers that create buggy code, the fact that lacking full requirements results in implementation not matching expectations, etc.

Project Review meetings to quickly identify what was good, so those things can be repeated and what was bad and find a way to not do those again.

Hopefully, these give you a good start. Good Luck!

Share Improve this answer

Follow

answered Sep 17, 2008 at 4:03

not-bob

835 ● 1 ● 8 ● 23

0

It seems the consensus is that the emphasis should be placed on unit testing. Bug tracking is a good indicator of the product quality, but is only is acurate as your test team. If you employ unit testing it gives you a measurable metric of code coverage and provides regression testing so you can be assured you didn't break anything since last month.

My company relies on system/integration level testing. I see alot of defects being introduced because there is a lack of regression testing. I think "bugs" where the developer's implementation of the requirements deviates from the user's vision is sort of a seperate problem that as Dan and rptony stated is best addressed by Agile methodologies.

Share Improve this answer

Follow

answered Oct 14, 2008 at 15:46

Sephrial

1,207 ● 1 ● 14 ● 26

Better than concentrating on any specific level of testing (e.g. unit testing), it's better to have tests on all levels (unit, integration, system), because different kinds of bugs are revealed. – Edu Apr 26, 2012 at 5:54