Code Quality: how to measure developers performance? [closed]

Asked 15 years, 8 months ago Modified 5 years ago Viewed 4k times

22

Closed. This question is off-topic. It is not currently accepting answers.

Want to improve this question? Update the question so it's on-topic for Stack Overflow.

Closed 11 years ago.

Improve this question

I work for a software development company and we have around 100 people working on a product, 1/3 of these people are QA. Lately management wants to have a better way to rate individual programmers performance so the suggestion was to use bug reports as a measurement. The more bug reports on a developer the worse he is. This seems ill-advised for more reasons than I can tell e.g. it is a subjective way of measuring, developers work on different projects of differing complexity. In addition if QA is measured for the number of bug reports they generate there will be a lot of discussions about the validity of bug reports.

What would be a better way to measure developers performance in such a setting?

One suggestion would be to not use bug reports from QA as a measure and instead use bug reports from outside, like beta testers then when such public bug reports are issued also let QA be measured by it.

EDIT:#1 After reading some of your excellent responses I was thinking that the general problem with the above described metric is that it is negative reporting bugs - it doesn't encourage producing good quality code.

EDIT:#2 I think the problem is that it is two worlds. There are the non-programmers on one side who treat programmers as workers basically, they want metrics loc/minute preferably. Then we have the Programmers, who want to see themselves as artists or craftsmen, "please don't disturb me I am c-o-d-i-n-g" :) I don't think measuring quality can be done by metrics not without being counterproductive. Instead things how a person reacts to bugs, willingness to change, creativity and above all quality of work are important and but mostly not necessarily measurable.

process **development-environment**

edited Dec 10, 2019 at 16:46

community wiki
5 revs, 3 users 100%
AndersK

18 Answers

Sorted by: Highest score (default)

34

Share Improve this answer

Follow

edited Mar 11, 2019 at 7:54

community wiki
2 revs, 2 users 67%
Greg Hewgill

23

Trying to measure programmers performance with bug reports is a bad idea. However, so is trying to measure performance with virtually any other metric. No matter what you do, people will figure out how to game it and give you what you're measuring without giving you what you really want.

From one of Joel's other articles:

> Robert Austin, in his book Measuring and Managing Performance in Organizations, says there are two phases when you introduce new performance metrics. At first, you actually get what you wanted, because nobody has figured out how to cheat. In the second phase, you actually get something worse, as everyone figures out the trick to maximizing the thing that you're measuring, even at the cost of ruining the company.

Share Improve this answer Follow

edited Apr 16, 2009 at 2:58

community wiki
2 revs
Chris Upchurch

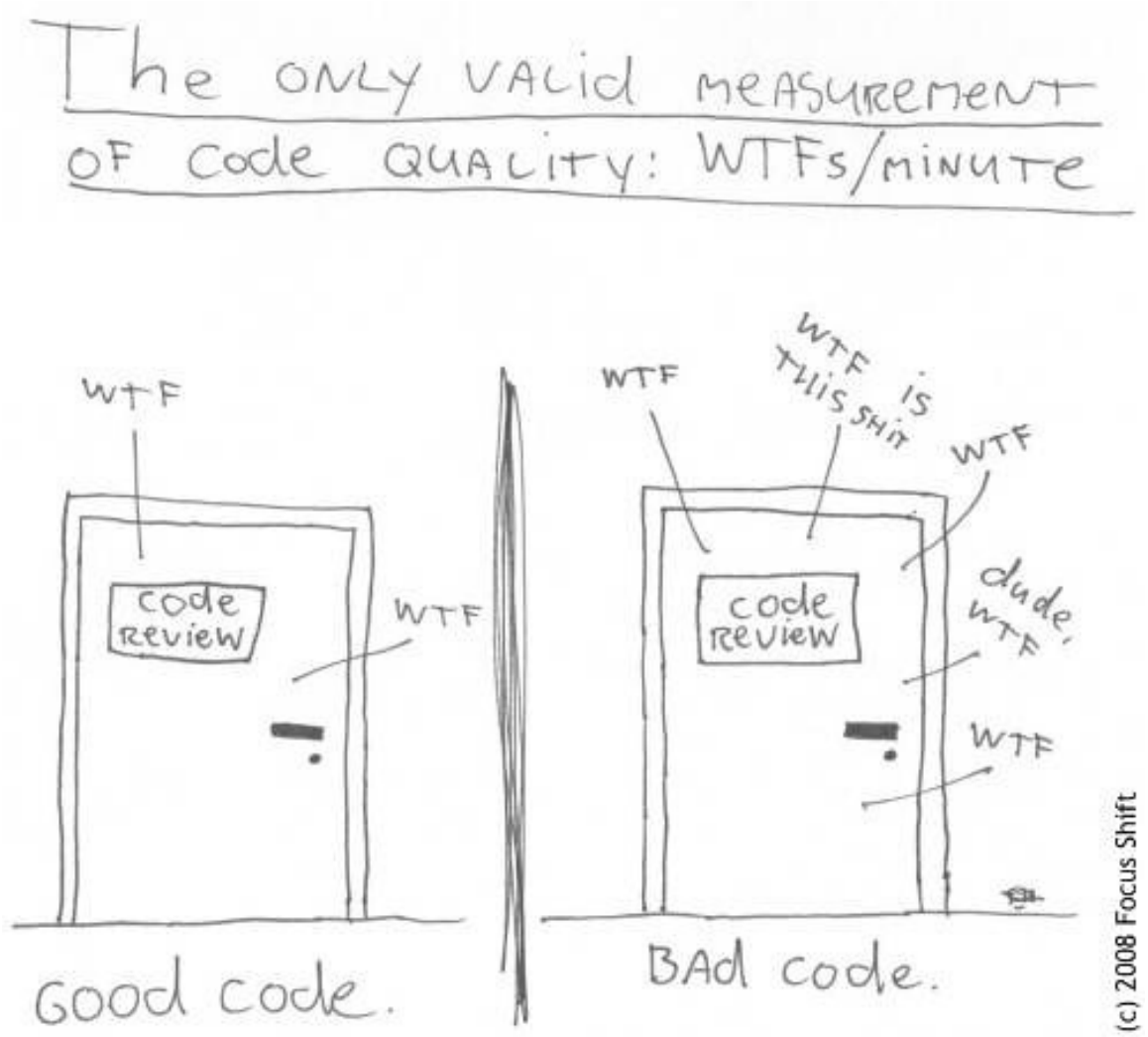
3 I disagree. Measuring a developer on their goals for the year, the milestones they achieved, as well as their overall improvement in skills is possible but not by numbers alone. A manager has to be involved. – ojblass Apr 16, 2009 at 2:53

3 I think you two are comparing apples to oranges – matt b Apr 16, 2009 at 2:57

2 I took Chris to be referring to raw numeric metrics -- number of bugs/lines of code/check-in's, etc ... they all say nothing. You've made a great point in that "A manager has to be involved." They also need to be able to understand/discuss code and help set legitimate measurable objectives. – Peter Meyer Apr 16, 2009 at 3:01

@Peter: Yep. The OP was asking about that kind of numeric metrics, which are almost always a lousy idea.
– Chris Upchurch Apr 16, 2009 at 3:09

22



Share Improve this answer

Follow

edited Feb 8, 2017 at 14:11

community wiki
2 revs
zalew

1 The second case would probably lead to better code, ultimately :) – user10996 Apr 21, 2009 at 3:58

The fundamental problem that I have with this type of rating system is that you end up with your team in competition with each other, rather than cooperating with

one another. What would be the incentive to work on hard parts of the code if you knew that you might pay a penalty? Just pick the easier things that are less prone to errors. Why help your colleague improve their code when doing so benefits them and potentially harms you with respect to the rating system.

I think you are better off using peer pressure to increase code quality: no one wants to write crap and no one wants to be known for writing crap. Make a real effort to drive defects down with TDD -- or at the very least with unit testing. Move to continuous integration and publicize who breaks the build. Make it the responsibility of the person breaking the build to fix it before they can create any new code. Things like this will drive quality up.

Once everyone is on board with the quality goals, rate the team, not the individuals. Make it a real benefit to work cooperatively. If you have slackers who are taking advantage of the team -- and everyone will know who they are -- work with them to improve and if they don't or can't, cut your losses and find someone who fits better with the team. If you have the right people, it probably will never get to this point. If they're the wrong people, both you and they are better off knowing it and moving on to a better fit.

If someone on the team really goes above and beyond, reward them with something extra, but make sure it really was an extraordinary effort beyond the rest of the team. If that's the case, the team won't mind (too much) because

they'll know that their shared reward was in large part due to that person's effort.

Obviously, all of the above should be taken as general rules. Although they like to call it management science, it's really more of an art. Understanding your team's dynamics is complicated business, but I think the general rule ought to be to encourage and reward teamwork.

Share Improve this answer Follow

answered Apr 16, 2009 at 2:56

community wiki
tvanfosson

5

The big problem I see with bug reports from the field is that a programmer may have programmed 100% to the specifications he was given and then the problem in the field was due to poor or incomplets specifications.

Let me give you an example: You develop and test an application on Windows Vista 32 bit and then it fails at a coustomer site where they are running 64 bit WIndows XP. Was that the programmers fault (especially if you never gave him a machine running XP 64 bit to test on)?

Once you realise that a bug can arise for lots of reasons, only some of which the programmer has control over, you need to be extremely careful that you do not setup an environment that leads to finger pointing and deviciveness. All members of the team need to be

working together to make the product better, not spending their day trying to assign blame for bugs to someone else.

Whatever you do, don't create an incentive system where someone gets bonus points for proving it was someone elses fault. Bugs need to be seen as owned by the entire orginization.

Share Improve this answer Follow

answered Apr 16, 2009 at 2:47

community wiki
JonnyBoats

3

That's a horrible metric, for the reasons you mentioned.

In addition, "outside" bug reports are also an unfair and unreliable way to judge developers - what if some developers work on an area of code that is used more than others? If my feature is used my 80% of users and your is used by 1%, who do you think is going to see more bug reports?

Any metric which can easily be gamed - or which gives other people being measured *incentives* to game them - are also horrible.

Share Improve this answer Follow

answered Apr 16, 2009 at 2:43

community wiki
matt b

3

Bug reports by developer is a horrible metric. As a QA Lead I have argued against this time and time again. Bugs **will** happen. How they are dealt with when they do is more meaningful.

A better metric is what the developer's bug reopen rate is. In other words, when QA logs a bug, which is then fixed, is the bug fixed correctly, or is there something that was missed, causing QA to reopen the bug?

The more often this happens, it is a clue that the developer may not be paying real attention to the problem. This is assuming, of course, that the bug was intelligently logged in the first place, preferably with steps to reproduce, the actual outcome, the expected outcome, and why. Screen shots help, too.

Obviously, this is just one metric on which to report. Other possibilities are

- Does the developer meet promised deadlines?
- Responsiveness to client concerns.
- Accuracy of any required documentation.

and probably others.

Edit: I have done both development and QA and was lucky enough during my development time not to have bug counts used against me in reviews. I argue against it at my current company in my current role because it is an unreliable metric IMO. We used reopen rate as a compromise to make upper management (read "pointy-haired" dev director) happy that they have something to report on. I am not a manager and don't actually generate any reports myself (in case that's the cause of some downvotes).

Share Improve this answer Follow

edited Apr 16, 2009 at 4:02

community wiki
2 revs
ssakl

+1. I'm leery of quality metrics in general, but I have to admit, this is a good one -- *if* it can be tracked accurately. (That's a big "if", but it doesn't invalidate the concept.) – Dan Breslau Apr 16, 2009 at 3:00

@Dan - We track this metric using reports from HP Quality Center. We keep a lot of history on status change and developer fields. Those without such a product may have a harder time generating accurate metrics. – ssakl Apr 16, 2009 at 3:23

There is still the problem of differentiating between coders who have to develop something very complex with those who have a simple task. – AndersK Apr 16, 2009 at 3:27

Still I find this reopen metric better than just a pure bug count. – AndersK Apr 16, 2009 at 3:29

This is not good, sometimes it could be an issue of QA and developers having different understanding of what the outcome is supposed to be, should the developer be punished for that or the manager? – Bjorn Apr 16, 2009 at 3:34

2

Well said by Chris.

We had a similar problem in our office. The CEO and other big wigs didn't know how to measure dev quality and they implemented their own rediculous measurements. Totaling a developers bug count is definitely not the measurement to go by. I don't know if there is a perfect answer but I would hope that my work is measured by whether or not I meet my deadlines and consumer feedback (are they happy with the product).

Share Improve this answer Follow

answered Apr 16, 2009 at 2:51

community wiki
dkpatt

2

We're professionals, like a lot of people. We also believe that we are artists, and in my opinion we are. Unfortunately (most) programmers are artists with a patron.

By saying that there's no viable metric to measure us is to say "just leave us alone and we'll do our job". That *may* apply to you, but how many coworkers have you had that you just wish you had a number to show how crappy they are? Subjectivity is nice and makes us all feel better, and creates a nice salary for the manager, but we *do* need some measure of programmer proficiency.

If we ourselves don't come up with something that makes management happy, then they will do the same thing as art patrons do. "I don't like it, you're fired".

World > Company > Product > Developer

As for a particular metric, I'm as lost as everyone else. The best I saw was the reopened bugs metric.

Share Improve this answer Follow

answered Apr 16, 2009 at 5:20

community wiki
Darren Clark

1

The only real way to validate quality is to review work... and to measure what is reviewed, and the amount of re-work. Bugs are **after the fact** ways of measuring this --- and only one metric, but reviews during development are better. Have a look at some of the metrics that TopCoder uses.

Share Improve this answer Follow

answered Apr 16, 2009 at 2:43

community wiki
Peter K.

yes from a programmer perspective I think you are right, but management wants measurable quantities and when you have a very large team with lots of developers its hard to produce any metrics for them using code review, it becomes highly subjective. – AndersK Apr 16, 2009 at 3:21

Reviews are just one metric TopCode users and those are based on very objectively-stated questions (the answers to which can be appealed). Another metrics that TopCoder uses to judge people are reliability: If someone signs up for a job, do they complete it. Management is quite interested in on-time delivery, I believe. – Peter K. Apr 19, 2009 at 21:44

1

Not only are bug reports a suggestive measure but they are not really comparable. How "large" is the bug? One big bug may be worse than having 5 small ones... on the other hand it may not be. So you would need to get into the specifics of each individual bug (which is going to be a nightmare).

You could go with how long it takes to fix each bug, but that can be easy to play - add a simple bug, fix it quickly, which counteracts the time it took to fix an honest to goodness bug that was harder to fix.

You can use lint tools and unit tests to improve the code quality without being punitive. The lint one is a relativley simple process change that you work at over time (start off with X warnings on an existing code base and then reward the people that remove the most warnings over a period of time - turn it into a positive rather than a negative).

Unit tests are another thing, reward the code with the fewest bugs and the most tests (if the tests are properly written then odds are the best tested code will have the fewest bugs). Again this is a positive reward and an encouraging thing for developers. The tests make the code better, and people are not penalized.

There are many other things like this, but off the top of my head those will have a noticeable impact (and are impartially measurable) on the quality of the product - which is the end goal.

Share Improve this answer Follow

answered Apr 16, 2009 at 2:44

community wiki
TofuBeer

1

I disagree with the "Measure by Bug count" concept even the tester is inside or outside.

Instead of directly counting the number of bugs, you can use some rating mechanism concerning the severity i.e I

mean to measure the carelessness of the programmer.

Say a programmer is writing a code without handling the exceptions. This problem is bigger than a error in complex logic in a complex algorithm. So using a rating mechanism for each bug would be better for such a scenario. In that case each error/mistake/bug will get a fair weight and we can get a Overall idea about the performance using the Sum of the error ratings..

In the other hand this approach also can make problems because the rating is also done by human beings. But having a team for make this rating will reduce such problems and the mechanism will get more usable with the time with necessary improvements and alternations.

But its your duty to sub group the bug categories and assign them the necessary weights.. I think you cannot do this at once. This "Definition" also should get matured over time with amendments.. :)

Share Improve this answer Follow

edited Apr 16, 2009 at 2:57

community wiki
2 revs
Chathuranga Chandrasekara

I recommend your management read *Implementing Lean Software Development: From Concept to Cash*, by Mary Poppendieck and Tom Poppendieck. They highly

1

discourage the idea of rating developers based on metrics. Their preference is to reward teams, not individuals.

Where that method isn't considered practical, I'd recommend (and so do they ... I think ...) peer reviews. Don't couch it in blunt terms like, "How do you rank your teammates?", though. Ask instead: Who do you go to when you have a problem you can't solve? Who has provided the best creative input to the project? etc. The answers to these sorts of questions can provide much better guidance as to who's putting the most into the team.

Share Improve this answer Follow

edited Apr 16, 2009 at 3:07

community wiki
2 revs
Dan Breslau

1

This metric stinks and will encourage really bad practices and infighting as to who caused which bug. Any metric of fault should be counterbalanced by a metric measuring success. People who write more code will by definition have more opportunities for mistakes. Depending on the available data you may wish to normalize your rating system. If one developer implemented one feature with no defects I would not rate him in any way better than a developer that implemented 243 features with 3 defects.

Rating developers requires management to put aside the numbers and observe each team member. An actively engaged manager will understand which developers have deficiencies and will work with them to improve their performance. This actually requires work by the managers to help each individual set and meet goals.

Share Improve this answer Follow

edited Apr 16, 2009 at 3:24

community wiki
2 revs
ojblass

1

I have three things to say about this:

1) a manager who suggests that "higher bug count == worse developer" or "... == better tester" may be a bigger problem for your company than any single developer could ever be. How did this person get to be part of the discussion about evaluating developer performance? If they're managing developers, they should know better.

2) The easiest way for a developer to game **any** metric related to bugs (bug count, reopen rate, normalized or not per feature/LOC/whatever) is to make their implementation as difficult to test as they can. Impossible to test means zero bugs found by QA. Also probably impossible to use, which means zero bug reports from the field (well, maybe one). Any bug count metric is an

incentive *against* testability. Ask management if that is really what they want.

3) If they ever actually implement this policy, run like hell.

Share Improve this answer Follow

edited Apr 16, 2009 at 5:00

community wiki
2 revs
Zac Thompson

I personally like the reopen rate, if anything, and I see this post in a user vacuum. Personally I write apps that take money, and provide a service. No matter how complicated I make my code, if I take your dollar, and then crash, I WILL get bug reports from the field. – Darren Clark Apr 16, 2009 at 5:08

... so then the developer is encouraged to make sure that the system is incapable of taking the dollar. Or to write a crash-proof "brick". Either of these approaches will reduce bug reports. Yes, I exaggerated deliberately, but I stand by "bug count metric is an incentive against testability".
– Zac Thompson Apr 16, 2009 at 5:59

1

After reading some of your excellent responses I was thinking that the general problem with the above described metric is that it is negative reporting bugs - it doesn't encourage producing good quality code.

That is true. And you'll have the same trouble with any other metric you apply. The key issue is that quality is not something we know how to measure numerically. Plus, you shouldn't really care about the question of code quality primarily if you're doing your job properly. Your real question should be, "How is this person helping us make money?"

Evaluation is not something you can do with numbers, but it is something you have to try to do. The best advice I can give you is that your managers simply have to work with the programmers and understand what the programmers are doing. Another important source of information comes from a programmer's peers who work with them day in and day out. Since we don't have a numerical way to measure quality, you will to some degree or another have to rely on the softer science to get insight into how well your programmers are performing.

answered Apr 16, 2009 at 5:34

community wiki
James Aguilar

Bugs found by QA = good. Bugs found by customers = bad.

1

If you need to measure developers, use #bugs found AFTER testing i.e. in Production/Release/'on the disc' code.

Same metric for QA... 'one team one dream'.

Michael

Share Improve this answer Follow

answered Apr 24, 2009 at 4:16

community wiki
Michael Dausmann

0

Before implementing any sort of metrics ask yourself.... ultimately... what is it you want to measure ?

-Programmer productivity are you not listening ?? duh

-Yeah sure.. but why is this important ?

Involving humans in metrics will innevitably try to optimize it for their own goals, thus, knowing this, you should be able to use this optimization in your advantage.

- Ask yourself then if someone optimize the metric, what will they be concentrated on ?

Then direct your metric in measuring someting that will positively effect the system, thus instead of measuring bugs per programmer which only gives amunition to

management to fire if things go bad, try to measure where and why the bugs occur, not who made them.

thus, bugs per modules, bugs per versions, bugs per code fix, bugs per feaures would be much more productive metrics and will help identify hotspots. Of course you can always tie it to someone as there is always an indirect link to a programmer, but before you place a programmer on the forefront of the blame war you better be darn sure HE-SHE is the cause.

- Ask yourself what kind of environment you want to create ? What will the reaction of the team, managers, directors will be when faced with the metric's publication ?

If you measure people, and make them look bad then you are asking for troubble. If you peasure product then the focus will not be on making themselves look good but making the product look good. This in turn will be a much better motivator and will foster positive team spirit.

- Make your metric public, any sort of information hiding will cause adverse reaction and injustice. Thus if you publicize your metrics be carefull on what they say.

Lastly if you really isist on measuring people then measure them all, programmer, architects, managers, salesmen, directors veryone should have the same scrutiny. Then hide the knives and place metal detectors on the doors. cause usually, transparency with people ina

company only works one way, from the top looking downwards.

Share Improve this answer Follow

answered Apr 16, 2009 at 5:19

community wiki
Newtopian

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.