Having to set objectives for developers, even though objectives don't work [closed]

Asked 15 years, 11 months ago Modified 12 years, 10 months ago

Viewed 82k times

86

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 12 years ago.

Improve this question

It is generally accepted that setting measurable objectives for software developers doesn't work , as too much focus on the objectives can lead to behaviour counter to the organisational goals (so-called "measurement dysfunction").

However, in my company, we are required to set objectives for all staff, and are encouraged by Human Resources to make them SMART. In the past, my fellow first-level managers (team leads) and I have tried a number of approaches:

1. Set measurable objectives that are additional to the normal job, like "Do training on technology X", "Create documentation for piece of code Y that no-one understands" and so on. When it comes to the annual performance evaluation, rate developers not on the written objectives, but rather on my opinion of the unmeasurable value of their normal work, since that is actually what the company cares about.
2. Set very specific objectives like "days' work done as recorded by the task management system", "number of bugs introduced", "number of production issued caused". This led to inflated estimates and incorrect classification of bugs, in order to achieve better "scores". Interestingly, even those developers scoring highly on this system didn't like it, as the intrinsic trust within the team was damaged and they didn't always feel they deserved their high position.
3. Set vague objectives that are variants on "Do your normal job well". When it comes to the annual evaluation, their rating does reflect performance against the objectives, but the objectives themselves are not measurable or achievable, which is frowned upon.

None of these is ideal. If you have been in a similar situation of having to create meaningful, measurable objectives for software developers in spite of the evidence against their effectiveness, **what approach has worked best for you?**

Related questions I found that don't quite address the same point:

- What are some good performance goals for a software engineer?
- Setting Performance goals for Developers
- What are suitable performance indicators for programmers?
- What is a fair productivity measurement technique for programmers?
- I need some career “Goals” for the next year

Update (18 November 2009): There are 10 upvotes for my question, and the highest-rated answers only have 4 upvotes (including one each from me). I think this tells us something: perhaps that Joel and the others are right, and that the combined wisdom of stackoverflow cannot come up with *any* compelling, measurable objectives for developers that could not be gamed without adversely affecting the true (unmeasurable) value of their work. Thanks for trying though!

evaluation

Share

Improve this question

Follow

edited May 23, 2017 at 12:09

Community Bot

1 ● 1

asked Jan 15, 2009 at 13:31

Paul Stephenson
69.3k ● 9 ● 52 ● 51

16 I prefer the DUMB methodology: "Do Ur Most Best." – Robert S. Jan 15, 2009 at 15:40

3 Lots of broken links. – crh225 Feb 16, 2016 at 21:15

Links are broken – Rodrigo Leite May 14, 2019 at 14:21

11 Answers

Sorted by: Highest score (default)

21

> what approach has worked best for you?

Only one objective: **pass a code inspection/peer review, with me as the reviewer, without me finding any bugs or having any other criticism, that has me asking you to redo something.**

Notes:

- I wasn't measuring new hires' ability to finish quickly, and didn't encourage them to: I wanted people to learn how to finish well (because if it's not finished well, then it's not finished)
- People learned what I looked for in a code review: so it's a learning opportunity **and** a quality control measure, and not just a management objective
- My comments would have two categories:

1. This is a bug: you must fix this before you check in
2. As a suggestion, I would have done such-and-such

- After a while, my reviews of a person's code would stop finding any "must fix" items (at which point I wouldn't need to review their work any more).

Share Improve this answer

Follow

answered Jan 15, 2009 at 14:07

ChrisW

56k • 14 • 120 • 233

Thanks, I like this. When I do review their code, I'm usually quite anal about not-so-urgent-but-in-the-long-run-important things like variable naming and code style. An objective like this might encourage them to get thinking along my lines more quickly! – Paul Stephenson Jan 15, 2009 at 14:11

6 I like this too, but it could lead to a blinkered pattern of development where everyone is just trying to please YOU at the possible expense of objectively best code. How many faults in your own approach have you fixed over the years, how many would you estimate remain to be ironed out? – Ed Guiness Jan 15, 2009 at 14:15

1 For me, variable naming and code style belong in the 2nd category; except if the style is *really* bad e.g. reusing one variable for too many difference purposes, I might say "you'll have to rework this because it's too complicated for me to review, I can't see 'by inspection' whether it's correct". – ChrisW Jan 15, 2009 at 14:19

Heh, obviously I know what is objectively best :-) . But yes, they might spend time pleasing my crazy quirks instead of doing more useful stuff. I reckon it would work better for

newer developers than the seasoned old hands.
– Paul Stephenson Jan 15, 2009 at 14:23

@edg: that's true about "blinkered" and "trying to please me", but their code also, of course, had to pass QA's black box system testing too. And, my judging whether or not *I* could maintain their code if necessary is quite an objective (not just subjective) measure, isn't it? – ChrisW Jan 15, 2009 at 14:24

14

Personally I try to set two sorts of objectives:

- Business-focussed objectives (this is why we are getting paid after all). For instance, "complete project X by 1 June 2009"). These objectives are often shared across several members of the team (and they are aware of this). The team can exceed the objective by bringing the project in early or by exceeding the functionality required. Individuals can exceed the objective by producing more functionality, having fewer bugs against them, or coaching and supporting other members of the team.
- Personal growth objectives, for instance completing a project involving a technology that the developer wants to add to their skill set, understanding the user's domain better, gaining leadership experience etc.

I feel that it is important that:

- Objectives are SMART

- Objectives are aligned with the needs of the business
- You do include "normal work" in objectives, in fact these are the most important objectives!
- The employee has some opportunity to exceed the objectives you set

Finally, I would stay away from software metrics as objectives - they are too easy to game and probably won't give you what you need. I would only use a metric where I want to coach someone in or out of a particular behaviour.

Share Improve this answer

Follow

answered Jan 29, 2009 at 21:51

Bids

2,452 ● 18 ● 26

9

This all boils down to the fact that "first level management", and most any management doesnt know their employees. Instead of being part of the actual day to day planning and development, things like SMART pops up. If managers were to spend more time with the guys who does the actual work, none of this would be needed.

Personally, I prefer working in an agile environment where it is *obvious* who of the developers performs in terms of productivity and quality awareness. A true agile approach requires that not only developers, but designers, testers, customers and product managers are

included in the process. This naturally leads to better insights from the managers point of view.

Share Improve this answer

Follow

answered Jan 15, 2009 at 13:41

Martin Wickman

19.9k ● 13 ● 84 ● 109

1 I'm basically a team lead, and I am part of the actual day to day planning and development. I also think that each developer's performance level is "obvious", but it is my subjective opinion and it doesn't fit in with objectives, so they become pointless. I'd rather we could ignore them altogether! – Paul Stephenson Jan 15, 2009 at 13:47

The point is that you cannot get any scientific measurement here, thus trying to do that is doomed and you should spend you time somewhere else imo. martinfowler.com/bliki/CannotMeasureProductivity.html has a piece about it. – Martin Wickman Jan 15, 2009 at 14:02

8

Measurable objectives I have seen so far:

- Pass a certificate exam
- Research technology X and hold a presentation about it
- Number of build breaking changes committed
- Number of wiki articles written on the internal knowledge management

How about asking your developers directly if they have some ideas for personal development which then could

be used for objectives?

Share Improve this answer

Follow

answered Jan 15, 2009 at 13:43

Petteri H

12.2k ● 12 ● 67 ● 96

1 All except breaking the build are my approach 1: what happens is that the good developers say "I was too busy doing that critical project you asked me to work on, so I didn't do the presentation or write the article". I can't penalise them for this so the objectives become meaningless.
– Paul Stephenson Jan 15, 2009 at 13:49

1 ditto what Paul said, and I'd have a problem with something as ephemeral as writing wiki articles - were they any good? are they still there? what about editing contributions? did they even have spare time for this? – annakata Jan 15, 2009 at 14:08

8

> Having to set objectives for developers, even though they don’t work

If your developers don't work, perhaps some objectives are *just* what they need to give them some incentive? ;-)

Share Improve this answer

Follow

answered Jan 15, 2009 at 14:26

Andrzej Doyle

104k ● 33 ● 191 ● 231

3 +1 for humour: I did wonder if it was ambiguous, but decided only if you deliberately misunderstood :-)
– Paul Stephenson Jan 15, 2009 at 14:31

2 Note that someone changed the title to "even though they (the objectives) don't work". I've since tightened it up to just "even though objectives don't work". Just adding the comment for anyone confused by this answer!
– Paul Stephenson Nov 18, 2009 at 15:46

7

"Ensure that at least n% of your code is tested by a suitable unit test" Use a coverage tool to prove it, and have someone else review for test quality.

Share Improve this answer Follow

edited Jan 15, 2009 at 13:45

answered Jan 15, 2009 at 13:35

Ed Guiness
35.1k ● 16 ● 113 ● 149

1 Define "exercised". If your just using coverage tools, its easy to get the code to execute without actually exercising it.
– Kent Boogaart Jan 15, 2009 at 13:39

@Kent, I meant exercise == execute. Could you expand how executing is not exercising and I'll gladly update my suggestion – Ed Guiness Jan 15, 2009 at 13:42

Sure. Just write a unit test that calls your method but does not make any assertions about the results of the call. You've executed the code - thus yielding higher code coverage -

without actually proving it is functionally correct.
– Kent Boogaart Jan 15, 2009 at 13:44

Thanks, something like this might be workable, as unit testing will become integral to their development and maintenance work. There may be issues with people writing valueless unit tests to meet the objective when they could be doing more useful work, though. – Paul Stephenson Jan 15, 2009 at 13:58

Agree, there's probably always going to be ways to game any specific measurement, which kind of reinforces the OPs point. – Ed Guiness Jan 15, 2009 at 14:13

4

I think that having very specific goals up front, i.e., SMART (maybe we work at the same place actually), seems like a good idea in practice but it isn't very practical for most teams.

The problem really is our incremental goals change. The business changes and as developers we need to react and react properly and in a reasonable time frame.

Consider setting goals that tie with your team or group's purpose in the organization. Your team wouldn't be funded if it didn't serve a purpose - a macro purpose. Have collective goals that exist across your entire team and that align to the business. Give people responsibility and hold people accountable. Celebrate their successes and failures (if we don't fail at times we're likely not trying and that's what you want from people). HTH

Share Improve this answer answered Jan 15, 2009 at 13:50

Follow

Eric Sabine
1,165 ● 7 ● 8

3

We have a number of metrics that are collected as programmers do work, such as:

- Number of SLOC changed / added
- Number of errors / bugs injected in various stages of the process (during peer review, post peer review, post release)
- Change requests fulfilled / rejected
- Formal documents (software version descriptions, design docs, etc.)

All of these are tangibles which I find useful in presentations for management and software quality assurance. But I have never found them terribly useful in actual evaluations of people's performance - which is the point made by several of the links you listed. I've found that Joel's points here are valid - metrics never promote a good team atmosphere.

Unfortunately, we all live in a world where metrics are required by others (management, quality assurance, outside contractors, etc.). I've found that a balancing act is required - providing those metrics, but also providing evidence of intangibles - intangible being what each programmer has accomplished that is not necessarily tracked. For example, I had a programmer who spent a large amount of time investigating legacy code that no

one else wanted to touch. Even though his metrics were low for that period of time, that effort was invaluable.

The only way I've found to include such things has been to push for the creation of an additional intangible category and give it equal weight with the other metrics. Usually this is sufficient to swing the balance for a particular programmer.

Share Improve this answer

Follow

answered Jan 15, 2009 at 15:28

Matt Jordan

2,884 ● 1 ● 27 ● 28

2

One of the problems would seem to be that as a division/department IT organisations don't have measurable strategic goals. If they did it would be easier to set the goals for the individuals.

e.g. If there was a departmental initiative to reduce the number of problem tickets raised, then, you could set an individuals goals based on the number of tickets related to the software they look after.

Since software development is largly a collabarative it would make more sense to set goals at the team level, and, then rate individuals according to thier contribution to the team.

Share Improve this answer

Follow

answered Jan 15, 2009 at 14:01

James Anderson

27.4k ● 7 ● 54 ● 80

1 +1 for setting goals only at a team level. Pinning each problem ticket on an individual is demotivating, destroys team spirit and often gives a skewed and inaccurate measure of the true situation, especially if the number of likely problem tickets is quite low (e.g. about one per developer).
– Paul Stephenson Jan 15, 2009 at 14:15

1

An objectives that I like is:

Solicit N positive reviews of your involvement in a project from the project client.

This helps as it is always good to have some written positive feedback from customers (internal or external). Its not hard to get, its relevant and it is an easy, but not meaningless tick on the list.

Share Improve this answer

Follow

answered Jan 15, 2009 at 21:47

Nat

14.3k ● 5 ● 43 ● 64

What if you work the whole year on a single project that has not been shipped to clients? 0 reviews. What if you work on a generic website with no set list of clients? – jmucchiello Jan 29, 2009 at 22:11

1 In both cases you are still working on a project that has clients, whether internal or not. You are soliciting a review of your involvement with the client, not a review of the project.
– Nat Jan 29, 2009 at 23:12

1

Determining how to align personal development with the projects being done is a key point in this I think. Having developers analyze themselves to find weaknesses along with giving feedback on others may be a way to find what may be improved and then finding a way to measure it. For example, I may find that I rarely document completed items and so on my objectives for the year I can state that I want to improve this and that the amount of documentation I produce can be a measure of that. It may work or it may backfire depending on how I follow it really. On the one hand there may be valid concerns for this being how I improve my work and do what may be viewed as the proper way while a passive aggressive or childish view would be to produce a mountain of documentation if it isn't that good in terms of quality as that can be improved next year as this can be another point to consider: What is supposed to be the motivation to improve as much as possible all in a year compared to spacing things out?

Defining an effective developer is another element to this. Is it the person that fixes bugs best? Does new work quickest? Does new work complete with tests and documentation even though it isn't done quick? What are you calling **effective** since the **"it depends"** standard response should be clarified here.

Share Improve this answer

Follow

answered Jan 23, 2009 at 16:56

JB King

11.9k ● 4 ● 40 ● 49