How long does it really take to do something? [closed]

Asked 16 years, 3 months ago Modified 16 years, 3 months ago

Viewed 1k times

4

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 5 years ago.

Improve this question

I mean name off a programming project you did and how long it took, please. The boss has never complained but I sometimes feel like things take too long. But this could be because I am impatient as well. Let me know your experiences for comparison.

I've also noticed that things always seem to take longer, sometimes much longer, than originally planned. I don't know why we don't start planning for it but then I think that maybe it's for motivational purposes.

Ryan

project-management time-management

Share

Improve this question

Follow

edited Sep 21, 2008 at 22:42

juan

81.8k ● 52 ● 164 ● 198

asked Sep 21, 2008 at 20:57

MetaGuru

43.7k ● 68 ● 195 ● 300

I think this question could use some detail. Can you give an example of a specific project and why you think it took longer than it should? – David G Sep 21, 2008 at 21:27

8 Answers

Sorted by: Highest score (default)

7

It is best to simply time yourself, record your estimates and determine the average percent you're off. Given that, as long as you are consistent, you can appropriately estimate actual times based on when you believed you'd get it done. It's not simply to determine how bad you are at estimating, but rather to take into account the regularity of inevitable distractions (both personal and boss/client-based).

This is based on Joel Spolsky's Evidence Based Scheduling, essential reading, as he explains that the primary other important aspect is breaking your tasks

down into bite-sized (16-hour max) tasks, estimating and adding those together to arrive at your final project total.

Share Improve this answer Follow

edited Sep 21, 2008 at 21:37

answered Sep 21, 2008 at 21:08

Dave Rutledge
5,535 ● 7 ● 29 ● 24

2

Gut-based estimates come with experience but you really need to detail out the tasks involved to get something reasonable.

If you have a spec or at least some constraints, you can start creating tasks (design users page, design tags page, implement users page, implement tags page, write tags query, ...).

Once you do this, add it up and double it. If you are going to have to coordinate with others, triple it.

Record your actual time in detail as you go so you can evaluate how accurate you were when the project is complete and hone your estimating skills.

Share Improve this answer Follow

answered Sep 21, 2008 at 21:03

Michael Haren
108k ● 41 ● 171 ● 206

2

I completely agree with the previous posters... don't forget your team's workload also. Just because you estimated a project would take 3 months, it doesn't mean it'll be done anywhere near that.

I work on a smaller team (5 devs, 1 lead), many of us work on several projects at a time - some big, some small. Depending on the priority of the project, the whims of management and the availability of other teams (if needed), work on a project gets interspersed amongst the others.

So, yes, 3 months worth of work may be dead on, but it might be 3 months worth of work over a 6 month period.

Share Improve this answer

Follow

answered Sep 21, 2008 at 21:14

brock.holum

3,213 ● 2 ● 22 ● 15

1

I've done projects between 1 - 6 months on my own, and I always tend to double or quadrouple my original estimates.

Share Improve this answer

Follow

answered Sep 21, 2008 at 20:58

Florian Bösch

27.7k ● 12 ● 51 ● 53

1

It's effectively impossible to compare two programming projects, as there are too many factors that mean that the metrics from only aren't applicable to another (e.g., specific technologies used, prior experience of the developers, shifting requirements). Unless you are stamping out another system that is almost identical to one you've built previously, your estimates are going to have a low probability of being accurate.

A caveat is when you're building the next revision of an existing system with the same team; the specific experience gained does improve the ability to estimate the next batch of work.

I've seen too many attempts at estimation methodology, and none have worked. They may have a pseudo-scientific allure, but they just don't work in practice.

The only meaningful answer is the relatively short iteration, as advocated by agile advocates: choose a scope of work that can be executed within a short timeframe, deliver it, and then go for the next round. Budgets are then allocated on a short-term basis, with the stakeholders able to evaluate whether their money is being effectively spent. If it's taking too long to get anywhere, they can ditch the project.

Share Improve this answer

Follow

answered Sep 21, 2008 at 21:12

Jason Etheridge

6,897 ● 5 ● 31 ● 33

Hofstadter's Law:

'It always takes longer than you expect, even when you take Hofstadter's Law into account.'

I believe this is because:

- Work expands to fill the time available to do it. No matter how ruthless you are cutting unnecessary features, you would have been more brutal if the deadlines were even tighter.
- Unexpected problems occur during the project.

In any case, it's really misleading to compare anecdotes, partly because people have selective memories. If I tell you it once took me two hours to write a fully-optimised quicksort, then maybe I'm forgetting the fact that I knew I'd have that task a week in advance, and had been thinking over ideas. Maybe I'm forgetting that there was a bug in it that I spent another two hours fixing a week later.

I'm almost certainly leaving out all the non-programming work that goes on: meetings, architecture design, consulting others who are stuck on something I happen to know about, admin. So it's unfair on yourself to think of a rate of work that seems plausible in terms of "sitting there coding", and expect that to be sustained all the time. This is the source of a lot of feelings after the fact that you "should have been quicker".

 answered Sep 21, 2008 at 22:41

Follow

Steve Jessop
279k • 40 • 469 • 709

0

I do projects from 2 weeks to 1 year. Generally my estimates are quite good, *a posteriori*. At the beginning of the project, though, I generally get bashed because my estimates are considered too large.

This is because I consider a lot of things that people forget:

- Time for bug fixing
- Time for deployments
- Time for management/meetings/interaction
- Time to allow requirement owners to change their mind
- etc

The trick is to use evidence based scheduling (see Joel on Software).

Thing is, if you plan for a little extra time, you will use it to improve the code base if no problems arise. If problems arise, you are still within the estimates.

Share Improve this answer

Follow

answered Sep 21, 2008 at 21:04

Sklivvz
31.1k • 24 • 118 • 174

0

I believe Joel has wrote an article on this: What you can do, is ask each developer on team to lay out his task in detail (what are all the steps that need to be done) and ask them to estimate time needed for each step. Later, when project is done, compare the real time to estimated time, and you'll get the bias for each developer. When a new project is started, ask them to evaluate the time again, and multiply that with bias of each developer to get the values close to what's really expects.

After a few projects, you should have very good estimates.

Share Improve this answer

Follow

answered Sep 21, 2008 at 21:15

Milan Babuškov

61k ● 49 ● 130 ● 180