Agile - Task Breakdowns - to estimate or not? [closed]

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

Viewed 3k times

4

Closed. This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

This question does not appear to be about programming within the scope defined in the help center.

Closed 7 years ago.

Improve this question

During our iteration planning, we frequently find ourselves in the same position as this guy - How to estimate a programming task if you have no experience in it

I definitely agree with prototyping before you can give a reasonable estimate. But the same applies to anything that needs a bit of architecture and design - but I'm not that comfortable doing all this outwith the scope of a sprint.

The basic idea is that you identify as many tasks as you can that you're confident of, and estimate these as normal. For those areas that you're unsure of then there

should be two 'types' of task identified: Investigation & Implementation.

Investigation tasks are brief descriptions of work that you're just unsure of, for example "Investigate how to bind Control X to data". An estimate is provided for these.

The Implementation task is a traditional rough guess, probably based on the story points assigned, of how long you think it would take to implement the feature.

During the sprint, when the investigation tasks have been completed, the developer should then be at a stage where they have a much better idea what is going on. 'Proper' Tasks can then be identified, which take the place of the Implementation placeholder. In addition, further Investigation tasks may be identified at this stage, and the cycle continues.

In the above example, we start with an Investigation task at 7 hours and an Implementation task estimated at 14. Once the first Investigation has been completed, Tasks 1, 2 and 3 will be identified and estimated with some degree of certainty, where Task 3 is another Investigation task from which Task 4 and 5 will be identified at a later stage. As you can see, the first Implementation estimate had delivery of the feature within 14 hours - but the reality is it took at least 4 + 7 + 3 + 4 + 2 = 20. A third more than the initial estimate.

alt text
http://www.duncangunn.me.uk/myweb/images/estimate.p

ng

All thoughts are welcome - my gut instinct is this will fly - am I right or am I the Wrong Brothers?

Cheers!

project-management agile estimation

Share

Improve this question

Follow

edited May 23, 2017 at 12:17

Community Bot

1 ● 1

asked Jan 8, 2009 at 22:50

Duncan

10.3k ● 14 ● 66 ● 96

3 Answers

Sorted by: Highest score (default)

3

What we do.

Some features involve new technology. We can't accurately estimate them. Period.

We make up a number. Based on a couple of things. How hard does it "feel"? Can we get by with some kind of "partial" or "just-enough" implementation?

- If it's hard, then it's hard. It will be expensive.

- If there's a lot of parts, with a kernel of goodness and some bonus stuff layered on, we have a possibility of putting just the kernel into a release, and setting other stuff aside for later. A very few things are "all or nothing" where a partial release isn't possible. In that case, we have to provide enough time for "all", and that gets expensive.

Our standard approach is to get stuff that works, and possibly defer things to a later sprint if we ran of out time because of unexpected complexities.

What you're calling "investigation", we call technical spike sprints. For stuff that's new, we make up estimate number to placate managers who feel it necessary to overplan things. Then we spike the technology. Once it's spiked, we can revise the estimates based on what we now know.

Share Improve this answer

Follow

answered Jan 9, 2009 at 11:58

S.Lott

391k ● 82 ● 517 ● 788

> Thanks, it's good to know that at least we're on the right track with this. I certainly like to spike solutions to problems where there is a great deal of uncertainty, before I feel comfortable giving an estimate. It's interesting to hear that sometimes you just need to put your foot down! – Duncan Jan 9, 2009 at 23:53

2

Actually, the implementation of the feature took 27 hours - you forgot the first investigation of 7 hours, so in reality the actual implementation took almost twice as long as the estimate.

There are two ways you can go on this:

1. Just make the estimate as best you can and potentially experience a blowout in your sprint and a declined project velocity (you should only do this if the feature is both urgent and critical); or
2. Schedule the investigation for this sprint and leave the implementation for another sprint - without an idea of how long the task will take, the Product Owner does not have enough information to make a decision about in which sprint to schedule it or even whether to do it at all. Only tasks that have been estimated should be included in your sprint.

The first choice means your sprint and project estimates are somewhat arbitrary. The second choice gives much more predictability to your sprints.

In your example, the initial investigation may be scheduled for Sprint 1 but without knowledge of how long the task will take the Product Owner can't decide how to schedule it. If you came back with an estimate of 200 hours the Product Owner may decide not to do that feature at all, or to delay it until Release 2 of the product. The estimate comes in and the Product Owner schedules Task 1, Task 2 and the investigation of Task 3 for Sprint 2.

After estimating Task 3, Tasks 4 and 5 can be scheduled in Sprint 3 or later.

Share Improve this answer

Follow

answered Jan 8, 2009 at 23:45

Chris Latta

20.6k ● 4 ● 65 ● 70

0

Estimating feature usually is complex task. After some time your estimation will become better. But good approach can be that you estimate features with the story points. Story point is abstract value (meaning agreed among the team) that express complexity of the problem. You should assign the same complexity (same number of story points) to the features of the similar complexity. Then later on it is enough to estimate only smaller set of features (or looking at the historical data) and you should be able to estimate how much time you need.

Features with the similar complexity need similar time effort for implementation.

Share Improve this answer

Follow

answered Jan 8, 2009 at 23:44

jan

273 ● 1 ● 6