How do you structure a development sprint? [closed]

Asked 16 years, 3 months ago Modified 12 years, 7 months ago

Viewed 5k times

6

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

So I have a backlog of features and we are about to get started on a sizable project. I am working on defining the structure of our sprints and I'm interested in the communities feedback.

What I'm thinking is:

- One day sprint planning
  - Fill the backlog and figure out what each dev will go after this sprint
- Three weeks of development
  - GO! GO! GO!

- Daily stand up meeting
    - Check to see if anyone needs help or feels off track
- Two days of sprint review
    - code reviews happen here, stakeholder presentations
- One day sprint retrospective
    - what did we get done in the last sprint? how can we do better next time?

Sprints should always end on a Tuesday (to avoid too much weekend stress).

Anything else? There is obviously more to agile than this. I want to provide the team with a simple outline of how we are going to operate as we get this project started.

agile

Share

Improve this question

Follow

edited May 5, 2012 at 18:54

Michael Myers ♦

192k ● 47 ● 295 ● 295

asked Sep 17, 2008 at 16:36

Eric Schoonover

48.4k ● 50 ● 159 ● 203

7 Answers

Sorted by: Highest score (default)

5

I'd consider experimenting with sprints that are shorter then one month.

Personally I find one-two week iterations more effective at getting effective feedback quickly. It also prevents any issues that may be causing problems at the iteration level building up to levels that become harder to manage.

Even for the 30 day sprint - two days sounds about a day to long for the sprint review... and one day sounds about 0.5 days too long for the retrospective. I've found that if you need much more than that there have been communication problems while the iterations has been going on - so you might want to look at needing long reviews as a possible red flag.

Of course that's just been my experience - of mostly developing web apps with smallish (4-12) person teams. You're experience may vary.

That said - I'd definitely give shorter sprints a try. Like integration builds - a lot of things get easier if you do them more often.

Share Improve this answer

Follow

edited Sep 18, 2008 at 23:36

answered Sep 17, 2008 at 17:21

user13165

1 I like the idea of multiple iterations within a given sprint.
– Eric Schoonover Sep 17, 2008 at 17:30

2

- Make sure the "stand-up" remains a STAND-up. It is very easy to slide into longer and longer meetings.
- One day of sprint planning and three days at the end may be too much. Only schedule as much time as you need.
- +1 to the idea of shorter iterations. Personally, four one-week iterations within a sprint have worked well. People are great at estimating near-term tasks; past that it becomes more and more guesswork.

Share Improve this answer

Follow

answered Sep 17, 2008 at 17:26

Jedidja
16.9k ● 17 ● 76 ● 114

1

Turn off email, cell phone and instant messaging apps for core code time. 10am to 1pm, 2pm to 5pm might be good blocks for this.

Order food, drinks for team when they are in "the zone".

Cancel all other meetings for the days of, before and after the planning session and the review days.

Share Improve this answer

Follow

answered Sep 17, 2008 at 16:41

sal

23.6k ● 15 ● 69 ● 85

1

Looks like a good approach. I second what adrianh and jedidja said about possibly shorter iterations. I like 1 weekers myself. As well as better estimation, it also keeps the idea of "working software" on a much shorter cycle.

A few questions:

Why are code reviews left until the end? Either pair program, or do your reviews as you go.

Does 3 weeks of development mean "dev, test, documentation, installers, etc" ? I.e. everything you need to be truly done?

Share Improve this answer

Follow

answered Sep 18, 2008 at 8:24

Paul Hammond

6,546 ● 3 ● 19 ● 10

3 weeks of development means everything we need to deliver the scope of work defined in that sprint (dev, test, documentation, pm work). Large formal code reviews are left to the end of the sprint but there will definitely be iterative reviews as we go along in the form of pair programming mostly. – Eric Schoonover Sep 18, 2008 at 9:51

0

We structure our sprints very similar to your outline except our sprint reviews are the last day of the sprint and generally on last about an hour. The sprint review is the time where you exhibit your work to the customers and any other interested parties, not the time to do code reviews. Code reviews, if you chose to do them, should be done periodically throughout the sprint. We used to have a one hour block each week where we'd go over developer nominated code, meaning we didn't waste time reviewing every LOC written.

We also end our sprints on a Tuesday and begin on a Thursday leaving Wednesday to wrap up loose ends and tackle technical debt created during the sprint.

Share Improve this answer

Follow

answered Sep 19, 2008 at 13:43

Mike Reedell

1,587 ● 15 ● 23

0

I don't recommend postponing code reviews until after the sprint, they should be an integral part of the development process. In other words, a task is not done unless the code has been reviewed (and tested, and documented, and ...).

Share Improve this answer

Follow

answered Sep 23, 2008 at 12:39

Anders Sandvig

21k ● 16 ● 61 ● 74

0

Its important to stay away from managing for the sake of managing. SCRUM only requires 1 meeting a day, and that's a short one. Additionally, during each sprint, the only other meetings are the Spring retrospective, and the sprint planning. This allows us to implement ROWE, or a Results Oriented Work Environment. Let your developers decide How, Where, When they will do thier development. Use your daily stand-ups to track that they are doing their work. Other than that, stand back and be amazed at thier productivity.

Ideas like "turn off cell phones, turn off IM apps, etc during coding" are all bad ideas. When you hire your team, you are hiring them with confidence that they know how to do thier job correctly. If you hired them with that understanding, why would you want to constrain thier ability to get thier job done the best way they know possible? If you're using SCRUM, then each developer will have chosen the work they feel they're able to do, your job as a Scrum-Master is to remove obstacles, not create them.

Code Reviews: Absolutely necessary. Peer reviews of code are a great teaching tool for junior developers attending meetings, and for the folks having thier code reviewed.

Design Documents: I personally feel that detailed design documents covering what the developer intends to do is very important, and I also feel they are an important part of the development process. Now, this is not specifically

in-line with agile development, but I personally regularly refer back to design documents created years ago to see what the original developer was thinking when they coded their modules.

Share Improve this answer

Follow

answered Sep 20, 2011 at 2:22

Mike Van

1,058 ● 7 ● 26