Which Agile software development methods have you had the most success with? [closed]

Asked 16 years, 4 months ago Modified 16 years, 3 months ago

Viewed 858 times

3

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 7 years ago.

Improve this question

There are numerous Agile software development methods. Which ones have you used in practice to deliver a successful project, and how did the method contribute to that success?

methods agile

edited Aug 26, 2008 at 18:52

Share

Improve this question

Follow

asked Aug 9, 2008 at 3:19

user290

5 Answers

Sorted by: Highest score (default)

7

I've been involved with quite a few organisations which claimed to work in an 'agile' way, and their processed usually seemed to be base on XP (extreme programming), but none of them ever followed anywhere near all the practices.

That said, I can probably comment on a few of the XP practices

- **Unit testing** seems to prove very useful if it's done from the start of a project, but it seems very difficult to come into an existing code-base and start trying to add unit tests. If you get the opportunity to start from scratch, test driven development is a real help.
- **Continuous integration** seems to be a really good thing (or rather, the lack of it is really bad). That said, the organisations I've seen have usually been so small as to make any other approach seem foolish.
- **User story cards** are nice in that it's great to have a physical object to throw around for prioritisation, but

they're not nearly detailed enough unless your developer really knows the domain, or you've got an onsite customer (which I've never actually seen).

- **Standup meetings** tend to be really useful for new team members to get to know everyone, and what they work on. The old hands very quickly slack off, and just say things like 'I'm still working on X', which they've been doing for the past week - It takes a strong leader to force them to delve into details.
- **Refactoring** is now a really misused term, but when you've got sufficient unit tests, it's really useful to conceptually separate the activity of 'changing the design of the existing code without changing the functionality' from 'adding new functionality'

Share Improve this answer Follow

answered Aug 9, 2008 at 3:39

Matt Sheppard
118k ● 46 ● 113 ● 134

6

Scrum because it shows where the slackers are. It also identifies much faster that the business unit usually doesn't have a clue what they really want delivered

Share Improve this answer Follow

answered Aug 9, 2008 at 3:21

SQLMenace
135k ● 25 ● 211 ● 225

4

Scrum.

The daily standup meeting is a great way to make sure things stay on track and progress is being made. I also think it's key to get the product/market folks involved in the process in a real, meaningful way. It'll create a more collaborative environment and removes a lot of the adversarial garbage that comes up when the product team and the dev teams are separate "silos".

Share Improve this answer Follow

answered Aug 9, 2008 at 4:52

Jeff Donnici
2,738 ● 19 ● 17

2

Having regular retrospectives is a great way to help a team become more effective/agile. More than adhering to a specific flavor of Agile this practice can help a team identify what is working well and adapt to a changing environment.

Just make sure the person running the retrospective knows what he/she is doing otherwise it can degenerate into a complaining session.

There are a number of exercises you can take a team through to help them reflect and extract value from the retrospective. I suggest listening to the interview with Linda Rising on Software Engineering Radio for a good introduction.

Do a Google search for "Heartbeat retrospectives" for more information.

Share Improve this answer Follow

answered Sep 24, 2008 at 21:59

colivier
621 ● 7 ● 11

1

I've been working with a team using XP and Scrum practices sprinkled with some lean. It's been very productive.

Daily Standup- helps us keep complete track of what and where everyone is working on.

Pair Programming- has improved our code base and helped remove "silly" bugs being introduced into the system.

iterative development- using 1 week iterations has helped up improve our velocity by setting more direct goals which has also helped us size requirements

TDD- has helped me change my way of programming, now I don't write any code that doesn't fix a broken test and I don't write any test that doesn't have a clearly defined requirement. We've also been using executable requirements which has really helped devs and BAs reach requirements understandings.

kanban boards- show in real time where we are. We have one for the Milestone as well as the current iteration. At a glance you can see what is left to do and what's being done and what's done and accepted. If you don't report in your daily standup something pertaining to what's on the board you have explaining to do.

co-located team- everyone is up to speed and on page with what everyone else is doing. communication is just-in-time, very productive, I don't miss my cube at all.

Share Improve this answer Follow

answered Aug 28, 2008 at 23:08

codeLes
3,069 ● 3 ● 29 ● 27