Agile in small bites: most bang for the buck [closed]

Asked 16 years, 1 month ago Modified 12 years, 7 months ago

Viewed 499 times

6

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

What single aspect of agile development should we implement first to improve our development process, and why?

I'm in a situation that's requiring me to "tweak" my process, rather than re-engineer it, and "agile" seems to be the mantra of the day. If we can make only one change that will improve something--quality, time to market, documentation, transparency, *etc*., what will have the most visible, positive impact?

If we choose correctly, we'll be able to make a second choice. :-)

Update: *What is your current SDLC?*
Environment: essentially "restartup." A *small* handful of developers; legacy products with 10^5-10^6 LOC and tens of thousands deployed worldwide; products are strongly interdependent; significant features added over the years, including many one-offs, w/o refactoring; tight schedules; superficial QA; no *post-mortems* or "process guru."

Typical process:

1. Create design/spec. Review by all stakeholders.
2. Code one or more features/fixes.
3. Revise design/spec to account for surprises.
4. Test features, record defects.
5. Prioritize new and remaining tasks.
6. Revise design/spec/schedule.
7. Return to Step 2 as necessary.
8. Release for beta, record feedback.
9. Return to Step 2 as necessary.
10. Official release.

Thanks for so many helpful suggestions and insights!

process agile

Share

Improve this question

Follow

edited May 2, 2012 at 0:36

QuantumMechanic

13.9k ● 4 ● 46 ● 66

asked Oct 25, 2008 at 17:47

Adam Liss

48.3k ● 13 ● 113 ● 152

See stackoverflow.com/questions/184025/… – S.Lott Oct 25, 2008 at 18:12

12 Answers

Sorted by: Highest score (default)

7

Iterative building

When we moved to having builds on a consistent basis (in our case weekly or twice per week) we saw the biggest improvement.

When every build was produced we sat down with the development team, the QA team and the product management team and created a list of the work that was included in the new build.

Everyone then helped answer the question of what should be included in the next build.

We have since added many other features of Agile development (including trying to implement a scrum to the letter), but nothing has given us as much "bang for the buck" as iterative building.

Share Improve this answer

Follow

answered Oct 25, 2008 at 18:10

Hortitude

14k ● 16 ● 60 ● 72

5

Iterative development. Work in small iterations (say 2 weeks), have 'ready' application by the end of each and single iteration, i.e. your testers should be happy to release the results to your customers.

This is the core. You could build on this.

Share Improve this answer

Follow

edited Oct 26, 2008 at 1:55

Mitch Wheat

300k ● 44 ● 477 ● 550

answered Oct 25, 2008 at 18:07

Ilya Kochetov

18.4k ● 6 ● 47 ● 62

4

I'm a big fan of mix-and-match, and an incremental change of the development process. I agree that iterative development should be your first goal, but I think you can approach it in even smaller steps.

From my experience, I would recommend the following order - pick the first you don't do already:

- *Fix Bugs First.* I wish I wouldn't have to say that. This is the call of sanity, and also required to have shorter cycles.
- *Small steps.* Train the habit of implementing the smallest change that is a visible step towards the next feature, then compile and test. Break down all your tasks into <1h units before starting to code. Aim for buildable, functional code at least every 15 minutes. This doesn't require much infrastructure change - except maybe fixing the incremental build and having fast machines.

Yes! Start with making sure developers have fast machines. How much better advise could get?!

- *Build Everything Daily.* Set up a double-click full builds from Source Control to installation medium, ideally on a separate PC. This are the first step to the frequent builds, but they help a lot on their own already. For us, it was a crucial step in getting reliable, reproducable build results.
- *Start writing Unit Tests.* Don't bother about coverage yet, don't enforce "write tests first", but put the framework in place. Write tests for new code and changes. Then run them with your daily builds.
- *Short Cycles.* Now it's the time, you have all tools in place to make weekly or two-weekly in house releases: The codebase is in a deliverable state many times a day, making the build is a double-click away, and at least something is working.

Share Improve this answer Follow

edited Oct 26, 2008 at 1:50

answered Oct 26, 2008 at 0:24

peterchen

41.1k ● 22 ● 108 ● 193

1 My take on your advice: 1. First, do no harm. Be sure the day's activity doesn't break yesterday's. If it does, fix it now. 2. Take small bytes. Change only a little at a time. 3. Validate early, validate often. Successful tests help measure progress toward the goal. Very helpful - thanks! – Adam Liss Oct 26, 2008 at 1:45

4

See Best concrete “how-to manual” on MANAGING Test Driven and/or Agile development?.

My recommendation is start with TDD first. It is very easy to do and has a profound impact on quality.

There are several parts to this.

1. Everyone has to get the tools (JUnit or whatever -- this can be hard in some cultures.)
2. Managers have to demand that the testing is done. They must never (NEVER) circumvent the unit testing. As soon as someone says "those tests don't matter, ship it anyway" you've undone all the good from TDD.

3. You have to manage by test case: how many written, how many passed. You have to define functionality via test cases: feature [X] has [n] test cases, of which some are done, some are in process.

Share Improve this answer

Follow

edited May 23, 2017 at 10:27

Community Bot

1 • 1

answered Oct 25, 2008 at 18:17

S.Lott

391k • 82 • 517 • 788

3

Agile is quite the buzzword now, but please keep in mind that it is **NOT** a silver bullet; it will not fix your development process just like that. You may want to read Steve Yegge's excellent article about Agile Development to balance the hype.

"Cherry-picking" some aspect of agile development may be difficult if you don't grasp the core of Agile. Agile is more than anything else **a way to think**: be flexible, accept that things will change, write code in short iterations focused on getting one or few features **complete**. The opposite of getting a single, complete, monolithic specification, writing all the code, documentation, and then ship it.

If you want to prove that Agile Development works, I'd probably vote for using the sprint to show what "release early, release often" means.

Share Improve this answer

Follow

answered Oct 25, 2008 at 18:09

JesperE

64.3k • 22 • 142 • 199

3

Make a cross-functional team with programmers, testers, technical writers and possibly sales/services folks all together. Make them realize the concept of 'done' i.e. something to be finished is something that is written, tested, documented, installed, deployed and ready for customer to use.

It is important because unless everyone from various functional areas gets together and focus on the single objective of getting something to the client, you cannot implement any other aspect of Agile framework.

Share Improve this answer

Follow

edited Oct 25, 2008 at 18:14

answered Oct 25, 2008 at 17:57

Ather

1,600 • 11 • 17

3

Totally depends upon your existing process, but I'll tell you that one of the best moves we made was to get the concept of the item backlog and daily 3-question (What did you work on since we met last? What are you going to work on today? What are the roadblocks keeping you from moving forward?) meeting in the morning to see

where we are and what we can do to move forward toward our short-iteration cycle endpoint.

It's good to be able to see the dynamic backlog of work to do and what is being worked on now and what's going to make it into the next iteration. It's good to be able to get an idea about where individual developers are and to help them eliminate any impediments to moving forward. It keeps developers from going dark.

Anyhow, that is my thought. It worked for us.

Share Improve this answer

Follow

answered Oct 25, 2008 at 18:28

itsmatt

31.4k ● 11 ● 102 ● 165

> This is certainly an opportunity for us to improve. I see immediate benefits from understanding where the difficulties are, and where the actual results differ from the plan. One day it may help us improve our ability to schedule our projects. :-) Thank you! – Adam Liss Oct 25, 2008 at 19:12

1

Start with unit testing if you're not already doing that. If you are unit testing, switch to test-driven development. These are both easy to add into existing processes and will pay immediate dividends. When you're ready to tackle process changes, bring in iterative development. If your current process is already iterative, then start doing frequent releases of your iterations to customers to get feedback.

If I had to sum up the "agile" way, I would say deliver high-quality business value early and often. The practices above will get you a long way along that path.

[EDIT] I guess what I'm suggesting is take an agile approach to adopting agile methods and start with easy things that deliver lots of value right away.

Share Improve this answer

Follow

answered Oct 25, 2008 at 18:17

tvanfosson

532k ● 102 ● 699 ● 798

I wouldn't say that starting doing TDD or even just unit testing is easy. When you have experienced developers working on a long-running project that is not used to it - and an existing architecture not designed with testing in mind - starting TDD is almost impossible. – Torbjørn Oct 25, 2008 at 18:45

This may be *exactly* what I'm looking for. Focusing on successful test completion, rather than simply "getting the code done," may shake things up just enough to get us all thinking about the *customer* again. Many thanks!
– Adam Liss Oct 25, 2008 at 19:09

Automated tests that run in an automated build.

1

Share Improve this answer

Follow

answered Oct 25, 2008 at 19:31

Paul Croarkin

14.7k ● 16 ● 81 ● 119

1

I would try to go with Test Driven Development. This will give you many things:

- You will get a pretty good unit test coverage (I'm not saying that unit test coverage is important).
- Developers will have more confidence that the code is really working (see * later for more information)
- You will be able to refactor the code more easily (because you have the tests).

---

[*] - Kent Beck in this area mentions Influence diagrams. In influence diagrams an arrow between nodes means that an increase in the first node implies an increase in the second node. An arrow with a circle means that an increase in the first node implies a decrease in the second node.

```
-----> [Stress] <--o-- / --o--> [RunTests]
```

The more stress you feel, the less testing you will do. The less testing you do, the more errors you will make. The more errors you make, the more stress you feel. Repeat...

How to solve this circle that leads to stressed developers not trusting their own code after a while?

Test first development changes the influence diagram:

```
[TestFirst] <--o-- / --o--> [Stress]
```

The more you do test first development, less stress you feel. The less stress you feel, the more tests first development you do.

This leads to better testing code developed by developers who trust their code.

Share Improve this answer

Follow

answered Oct 25, 2008 at 21:44

David Pokluda

11k ● 5 ● 29 ● 26

... not to mention fewer surprises, particularly as the deadline approaches. Very good advice - thank you! – Adam Liss Oct 25, 2008 at 21:47

1

In addition to all the good advices already provided and that I concur with, I'd suggest to strengthen QA with automatic, repeatable tests. Investing in automation will

allow you to be more confident when changing already deployed code.

Create regression suites for the new features at the same time as you implement them.

QA can use exploratory testing as an alternative to find holes in your development process.

Share Improve this answer

Follow

answered Nov 2, 2008 at 8:10

philant

35.7k ● 11 ● 73 ● 113

0

I think the two most valuable and easy to implement aspects of Agile are

1. daily stand ups - have a brief daily meeting with the team to review status. Use the 3 questions. Avoid cross-talk, chatter, and bitching. Keep it quick and on on-point.
2. time-boxed iterations - breaking the project up into two or three week cycles forces you to work towards manageable goals on reasonable deadlines

Share Improve this answer

Follow

answered Oct 25, 2008 at 21:04

Jason

88.9k ● 15 ● 136 ● 149