Agile

Alex Sverdlov
alex@theparticle.com

1 Introduction

Started with a "Manifesto for Agile Software Development", with basic ideas being:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile is generally the opposite of waterfall: it is iterative, with users and developers often being on the same team—with working-code created early in the development life-cycle before all requirements or design being complete.

Lots of new methodologies describe themselves as "agile". The basic idea of agile is to be: ...agile regarding development, and to tighten the loop between requirements, development and feedback from users.

To ensure rapid development, there is a need to minimize errors during coding—so unit testing is often used to verify that coding changes in one part of the system do not break other parts of the system.

Once code changes can be made reliably, it becomes feasible to quickly prototype solutions and have confidence that nothing major breaks.

With quick prototypes and changes, users can be looped in for quick feedback. Quick feedback identifies requirements and design problems early, making them easier to fix (by using more quick reliable changes), etc.

Another aspect is 'pair programming', where two developers are sitting shoulder-to-shoulder working on the same piece of code. This actually works well in many settings—as the ideas bounce around and code gets developed quicker.

Also, with quick round-trips between "changes" vs "feedback", it becomes feasible to build a project schedule around such few-week sprints (as opposed to months of development without feedback from the users).

There are quite a few issues with Agile that most Agile proponents don't like to mention: some creative steps often can't be decomposed into short-two-week cycles.

Often the task driven nature of sprints limits the ability of going off-rails and trying arbitrary things to see if they'd work better. The ultra-scheduled nature of projects can often choke off creativity. These are mostly artifacts of Agile for management reasons vs Agile for developers.