COMP33711 - Agile Software Engineering

Christopher Williamson

December 13, 2016

Contents

1 Introduction

'We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value items on the left more.'

1.1 Key Ideas for Agile Approaches

One of the key ideas is to avoid waste which commits people to delivering real value. Trust is also a key idea. This includes having a self-organising team, empowering developers to make key decisions and including the customer as a member of the team. Simplicity of both the process and product is important. YAGNI (You Aren't Gonna Need It) is a principle that is often followed which basically means that a programmer should not add functionality until deemed necessary.

Finally, feedback is another key idea. We should aim to 'fail fast' by making sure that failure is visible quickly. Short iteration, close customer involvement in development, team retrospectives, high-coverage automated test suites and pair programming are all useful idea that aid the feedback process.

1.2 12 Agile Principles

To help people gain a better understanding of what agile software development is all about, the members of the agile alliance refined the philosophies captured in their manifesto into a collection of twelve principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversations.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximising the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

1.3 N Agile Practices

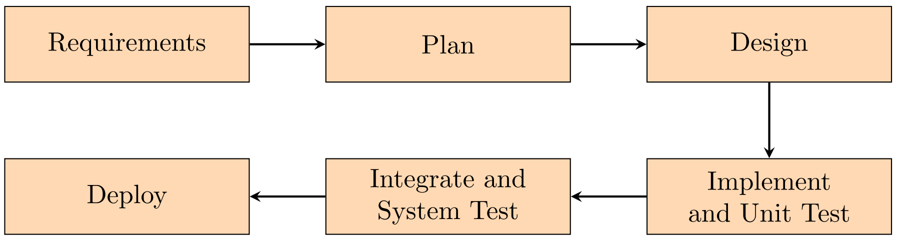
There is no limit on agile practices but some examples are short iterations, user stories, story points, planning games and TDD.

1.4 M Agile Methodologies

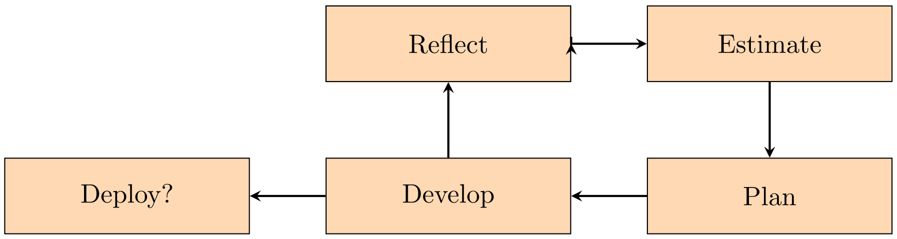
As above, there is no limit on agile methodologies but some examples are eXtreme Programming (XP), Scrum and Crystal.

1.5 Waterfall VS Agile

When using the waterfall approach, we usually have the following structure:



When using the agile approach, our diagram looks like this:

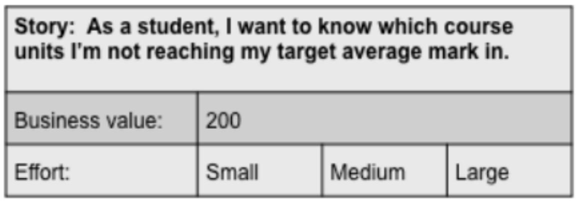


This diagram shows that the process is iterative and incremental as we loop back to the start (with a reflection stage) at the end of every development cycle (iteration).

1.6 Iteration Stages

1.6.1 Agile Practices: User Stories

Agile takes a different approach to requirements gathering. Instead of using a typical requirements specification, requirements are gathered as 'user stories'.



Story: As a student, I want to know which course units I'm not reaching my target average mark in.

Business value:	200		
Effort:	Small	Medium	Large

1.6.2 Estimation

You are given a set of user stories. Before starting to build, you need to decide what to commit to.

1.6.3 Planning

Next, you select the stories from your backlog that you will implement in the iteration. The key here is balancing business value achieved, effort expended, dependencies between stories and the amount of learning gained from implementation effort. Documenting the plan can be done by using a task board.

1.6.4 Development and Showcase

Next you implement as many of your selected stories as you can and show the customer what you've achieved. The customer will sign off the stories that have been completed satisfactorily and the rest will be put into the backlog for the next iteration.

1.6.5 Feedback

Finally we hit the feedback stage where we receive feedback from both the customer and the development team in the hope that the feedback will improve out process for the next iteration.