

Samuel Withers: 1809713

Agile Processes and
Methodologies

Introduction

During the early 1990's, it was common for software development projects to be cancelled halfway through due to the rapidly changing technology of the time. It was estimated that the average time between project conception and an actual product being produced was about 3 years [1], unfortunately the advance of technology was meaning a lot of projects were obsolete by the time they came to fruition. This was just the average; many projects went on much longer than 3 years meaning the problem was much worse. Because of this, a different business strategy was needed, something that was able to cope with changing needs during a project [2].

This led to the now-famous meeting at Snowbird ski resort in Utah in February 2001. The group consisted of like-minded software professionals all with one goal in mind, to change the way software development practices to make it faster and abler to cope with change. A way to build working software and get it to the hands of end users. From this meeting, the 'agile manifesto' was born. The manifesto states the four core values of Agile Software development [3], they are:

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

In essence, Agile software development is all about effectively communicating with the client and your development team to produce a product that perfectly fits the clients' needs, whilst adapting easily to any change that may take place along the timeline of the project [4] [3].

Agile software development has produced many branching methodologies, including, but not limited to, Scrum, Kanban, XP (extreme programming) and SAFe. Each of these methodologies follows the same core principles of Agile software development, but implement them in varying ways. Some companies follow one of these methodologies, others cherry pick the components from different methodologies that best suit the company [5]. For example, the DVLA uses the daily stand-up from Scrum, the planning board from Kanban and some of the software testing ideas from XP. This gives the company their own unique brand of Agile development that best suits the workflow of the company, whilst still maintaining the core principles of Agile.

In this report Agile development processes are outlined and the Scrum methodology processes discussed in detail. The validity of Agile processes in different types of companies is analysed and the appropriate settings in which to use Agile are discussed.

Iterative and incremental development

Agile software development differs from that of traditional software development in that it is both incremental and iterative, whilst traditional is only incremental. It is worth noting that certain traditional models, such as waterfall, are technically not incremental as they look at the product as a whole and not in parts.

Incremental development is where software is built and delivered in pieces, each increment can be as small or large as needed and represents a complete subset of functionality. It is generally assumed that an increment of work will not be revisited, as it has been fully coded and tested the first time.

Iterative development, on the other hand, makes progress through refinement. Meaning each iteration will be revisited and improved upon. A skeleton of the whole project is outlined with the first iteration and then improved upon in the next iteration. With each iteration then revising the previous.

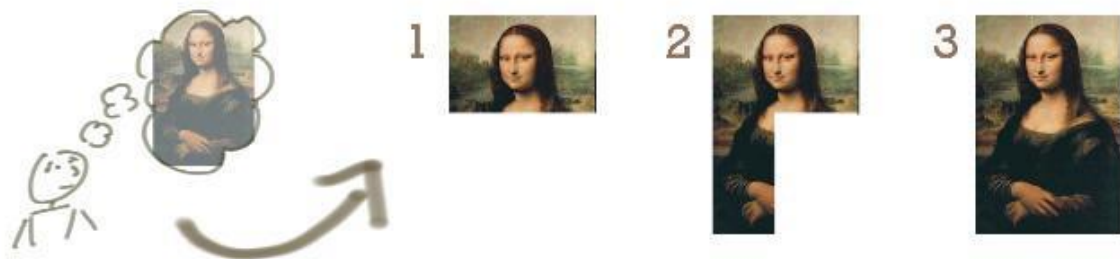


Figure 2: Example of incremental development, with pieces of the work being finished and then the next piece added [16].



Figure 1: Example of iterative development, with an outline of the whole project being produced and then revised with each iteration [16].

Agile uses a combination of both iterative and incremental development. This is done with the idea to get a working version of a feature out the door as soon as possible, however minimal the feature. The feature will be the iteration of the product, the first iteration in agile is usually called the MVP (minimum viable product), and the functionality of this feature will then be improved upon in an incremental fashion, making Agile development both iterative and incremental [6].

Scrum

Scrum is an Agile methodology, a way of managing a project using Agile principles. Scrum is probably the most well-known and commonly used Agile development methodology. It is often what people think of when they think of agile development. This is because it encapsulates all the major agile principles in an easy to understand way [7].

Scrum is typically conducted with small teams, and within these teams there are multiple roles which are important to understand before discussing the processes involved in Scrum.

Product Owner

The product owner is a member of the team who represents the voice of the customer [8]. It is their job to maximise the value of the product being developed, be that in terms of money or features. They are the sole person responsible for the product backlog, however, this product backlog will still likely express the views of others in the team. The product owner will always be held responsible for the product backlog.

Development Team

The development team is responsible for the development of the product. In Scrum this means building and testing increments every sprint cycle (typically two to four weeks). There are typically a number of developers within one scrum team as opposed to a single product owner [9].

Scrum Master

The scrum master is responsible for removing any impediments to the rest of the team throughout the project. They facilitate crucial meetings such as daily stand-ups and sprint planning. They exist to ensure the team can work smoothly [10].

Scrum Development Cycle

Sprint

A sprint is a standard development cycle in Scrum. It is typically a period of two to four weeks. Each sprint is typically one iteration of the product, with the development during the sprint being incremental. The sprint will start with a planning meeting to provide the sprint backlog. This is all the work to be done during that sprint defined in the format of user stories. User stories are a way of writing a requirement from a chosen perspective in plain and simple language. At a sprint planning meeting, everyone in the team will participate to choose which stories make up the sprint backlog, this means that everyone is on the same page with the work that should be done during that sprint.

Daily Stand-Up

The daily stand-up is a meeting that happens at the start of the day. This is a short meeting limited to 15 minutes in which each team member outlines what they have achieved the previous day, what they intend to achieve today and whether there are any blockers

preventing them from meeting their goals. It is time limited to ensure that it does not take time away from the working day, and it is called 'stand-up' as it is intended to be performed standing up, to increase productivity.

Sprint Retrospective

At the end of the sprint another meeting is held with the entire Scrum team. During this meeting the prior sprint is reflected on and ideas to improve moving forward are discussed. It is commonly conducted in a start-stop-continue format. This is where the team discusses what they should start doing in the next sprint, what they should stop doing from the previous one, and what they should continue doing as it worked well in the previous sprint [11].

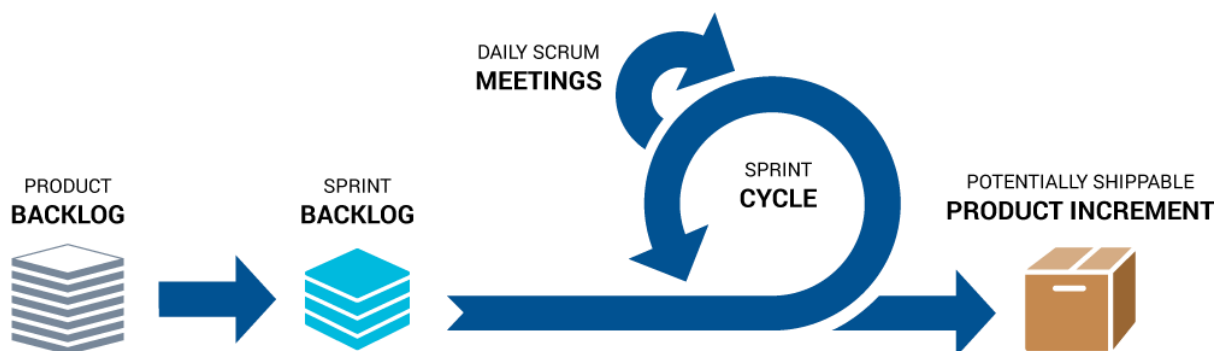


Figure 3: A visual representation of a Scrum cycle [15]

The image above shows one version of the Scrum cycle. It begins with the product backlog being defined followed by the sprint planning meeting to define the sprint backlog. It is worth noting that the product backlog can be changed or added to at any time, however once the sprint backlog is decided upon it cannot be changed for the duration of that sprint. The product backlog being flexible is one of the advantages of an agile workflow, it means that should the project need more functionality that originally thought, it can be added without affecting the workflow. Also, should certain features on the project be deemed unnecessary they can simply be taken out of the product backlog. At the end of the sprint cycle that 'increment' of the product should have working functionality meaning it is 'potentially shippable'. This is part of the reason Scrum is often useful, working software can be shipped out the door and this can be used as an easy measure of progress for the project backers.

When is Agile useful?

Agile was created to help combat a world where clients did not know exactly what they wanted from a product at its inception, and therefore were not happy with work that had been put in 6 months to a year down the line as their vision had changed since the start of the project. This led to many cancelled projects. Therefore, Agile is very useful in projects with changing goalposts, or projects where the design may change throughout. Agile allows the client the flexibility to sit down and talk with the team throughout the project, not just at the start, so if the needs of the project change, the team can easily adapt to these changes [4].

Agile is brilliant in environments where the workflow isn't identical day after day, as adapting to change is built into the way Agile is conducted. In a situation where the workflow is almost identical each day, there may not be that need for Agile principles and a more rigid structure may allow for a more efficient workflow. This means that Agile lends itself to projects where the teams within are trying something new or trying something in a new way. If it is something routine that has been done many times before, it is likely that Agile practices are not needed as much [4].

Projects with tight timeframes can suit an Agile workflow as if Agile is done correctly, time should be used very efficiently with very little downtime delaying work. Especially with the Scrum methodology, the daily stand-ups should allow the team to overcome any blockers to work as they come up, so they should not delay work for an extended period of time.

Case Study: Spotify

Spotify is a streaming service for music, podcasts, radio and a few other formats. It is primarily a music streaming service and can be thought of as "a magical music player in which you can instantly find and play every song in the world [12]". Spotify is a company that started small (in 2011 Spotify had only 30 employees) and has grown into a worldwide company with over 2000 employees across multiple countries [13].

The company has adopted an agile way of working since its inception and is an excellent example of scaled agile being done correctly and effectively. Spotify uses an Agile methodology very similar to Scrum, but with a large scale implementation. Spotify have given this scaling method the name "Spotify Tribe" [14].

The basic group in Spotify is called a squad, and it is almost identical to a Scrum squad. A Spotify squad is normally 6-12 people. Each squad is designed to have all the skills and tools necessary to go from planning to production and they decide on their own way of working. For many this is standard Scrum, for others it is Kanban or a combination of agile methodologies. Each squad decides how to work so they can optimise their own workflow without the constraints a large corporation may have on a small group's workflow. Each squad is given a long term goal to work towards, this allows a squad the time to become experts in a certain field, whether that be UI, radio, search queries or something else.

Multiple squads working on related features are then grouped together in a “Tribe”. Typically consisting of 50-100 people. A tribe is a group of squads that work in a similar area. They are designed to be small enough so that everyone will be able to know each other, but not so large that the size of the tribe hinders the workflow. The tribe has a Tribe-lead who is responsible for creating and setting a good environment for the squads to work in [12].

Spotify recognise that squads may often be dependent on one another, so this is perfectly allowed, however Spotify do try to cut down on cross-squad dependencies as they generally slow down workflow. The ultimate goal is for squads to be as autonomous as possible.

Spotify does have a separate operations team to the development teams, however the operations team do not make releases for the squad. They are designed to help and support the squads to release the code themselves, allowing the squads to be more autonomous [12] [14]. This is based on face to face communication instead of extensive documentation, which is one of the core principles of agile.

One problem with the level of autonomy the squads have is that one squad may have the solution to a problem that another squad has, but this may never be communicated therefore wasting work time. To counter this Spotify has chapters. A chapter is made of members of varying squads with similar skills/roles within the same tribe. The chapter members will regularly meet to discuss any problems they are having in order to collaboratively solve said problems. A chapter has a traditional line manager, but the line manager is also part of a squad [14] [12].

One level higher than a chapter is a guild. A guild is organisation wide, independent of tribes or squads. There may for example be a ‘testing guild’ which includes all the chapters within that area. However, anyone can join a guild and attend events for that guild. Each guild will have a ‘guild co-ordinator’ who is in charge of the guild and organising events for the guild [14] [12].

Spotify as a company has grown almost exponentially since its creation, and has successfully scaled its agile processes with the growth of the company. It is one of the best examples of a large company using agile processes effectively and choosing and modifying the parts of agile that suit the company. This is what agile processes are all about, the ability to be flexible and transform the company through growth whilst still allowing the individual in the company to be autonomous and work in an environment that suits them.

Conclusions

Agile methodologies can be a brilliant way of working when they are implemented correctly. However, they may not work for every single company. Each company has to evaluate their own business needs and their employees before deciding on the methodology for them. It may be that a mixture of traditional and agile processes perfectly suits the company. Spotify is an example of a large company that has incorporated their own scaled model of agile processes. Incorporating elements from multiple methodologies whilst including their own elements where it suits their needs. This works for Spotify, but may not work in the exact same way for another company. However, companies that have constantly changing needs and are needing to adapt to different technologies/growth may find it beneficial to adopt some agile processes, otherwise they may risk being left behind by competitors who are able to adapt better.

References

- [1] "To agility and beyond: The history - and legacy - of agile development," TechBeacon, [Online]. Available: <https://techbeacon.com/agility-beyond-history—legacy—agile-development>. [Accessed 12 05 2018].
- [2] N. W. J. W. Roderick Chapman, "What Can Agile Methods Bring to High-Integrity Software Development?," *Association for Computing Machinery*, vol. 10, pp. 38-42, 2017.
- [3] "Agile Manifesto," 2001. [Online]. Available: <https://agilemanifesto.org/>. [Accessed 06 12 2018].
- [4] J. A. Highsmith, "Agile project management : creating innovative products," in *Agile project management : creating innovative products*, Boston : Addison-Wesley, 2004, pp. 13-21.
- [5] J. A. Highsmith, "Agile project management : creating innovative products," Boston : Addison-Wesley, 2004, pp. 1-12.
- [6] A. Ghahrai, "Difference between iterative and incremental development in agile," Testing Excellence, [Online]. Available: <https://www.testingexcellence.com/iterative-incremental-development-agile/>. [Accessed 30 12 2018].
- [7] C. Drumond, "Scrum," Atlassian, [Online]. Available: <https://www.atlassian.com/agile/scrum>. [Accessed 12 2018].
- [8] "What is a product owner?," Scrum.org, [Online]. Available: <https://www.scrum.org/resources/what-is-a-product-owner>. [Accessed 30 12 2018].
- [9] "What is a scrum development team?," Scrum.Org, [Online]. Available: <https://www.scrum.org/resources/what-is-a-scrum-development-team>. [Accessed 01 01 2019].
- [10] "What is a scrum master?," Scrum.org, [Online]. Available: <https://www.scrum.org/resources/what-is-a-scrum-master>. [Accessed 01 01 2019].
- [11] "Sprint Retrospective," Mountain Goat Software, [Online]. Available: <https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-retrospective>. [Accessed 01 01 2019].
- [12] H. K. & A. Ivarsson, "Scaling agile @ Spotify," 10 2012. [Online]. Available: <https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf>. [Accessed 11 01 2019].
- [13] "Number of Spotify employees from 2011 to 2016," statista, [Online]. Available: <https://www.statista.com/statistics/245130/number-of-spotify-employees/>. [Accessed 11 01 2019].
- [14] "Exploring Key Elements of Spotify's Agile Scaling Model," Kendis, 23 07 2018. [Online]. Available: <https://kendis.io/spotify/exploring-key-elements-spotifys-agile-scaling-model/>. [Accessed 11 01 2019].

- [15] "Agile Methodology," Stigasoft, [Online]. Available: <http://www.stigasoft.com/agile-scrum-methodology.html>. [Accessed 01 01 2019].
- [16] "Iterating and Incrementing to 'Get What You Need'," InfoQ, 23 01 2008. [Online]. Available: <https://www.infoq.com/news/2008/01/iterating-and-incrementing>. [Accessed 12 2019].