22F_TWR2000_450 Technical Writing I
Instructor Mary Preston

Assignment 3 Writing Descriptions and Instructions

Brent Marshall 040108068

November 28, 2022

Agile Software Development

Overview

Agile software development is process for creating software products in a highly iterative fashion. It is used by most professional software organizations today. The software development is organized around periodic **sprints**. Sprints occur at a regular frequency chosen by the team, usually one or two weeks.

Agile also mandates daily **scrums**, where the entire team meets. They discuss what they plan to achieve that day, and any issues they may have encountered. Scrums are to be kept to a maximum of 1/2 hour so they don't impact the productivity of the developers.

Finally, Agile advocates that requirements are managed using a **backlog**. The backlog can change at any time. Each requirement is documented in a **user story**, which provides all of the details of how it should work, and success criteria for declaring it complete.

History

Decades ago most software was built using the Waterfall development process which divided the entire duration of the project into phases: analysis, design, implementation, and testing. The idea was that these phases were organized like a waterfall: you can go down a waterfall, but not up, so you would never go "backwards" in the process.

Waterfall project nearly always missed their dates, and exceeded their budget as a result. This was for two main reasons:

- over the duration of a large project, requirements would continuously change
- during implementation problems were discovered that were not apparent during design

Both of these problems would result in costly unforeseen mid-project redesigns. This also violated the whole premise of the waterfall method.

Agile was invented to address these problems. The design happens continuously, allowing the team to address requirement changes rapidly, and react quickly to implementation issues. Hence the name "Agile".

In 2001, 17 prominent advocates of reforming software development processes got together and summarized their value system in the "Agile Manifesto", which appears in the table below:

We value this	Over this
Individuals and interaction	Processes and tools
Working software	Comprehensive documentation
Customer collaboration	Contract negotiation
Responding to change	Following a plan

In the following years, the founders and others went on to write many books describing and formalizing the rules of Agile. The approach evolved rapidly, but is now more or less standardized.

The essential elements of Agile are described in the sections that follow.

Team Composition

The roles within the development team have standard names and responsibilities.

Product Owner

The Product Owner (or PO) is responsible for ensuring that the software created meets customer requirements. He or she meets with customers, and reviews general market trends to understand what is needed. Then the PO will capture the requirements as user stories in the backlog.

The PO must be accessible to the rest of the team if they have questions.

Once the software is ready, the PO has the final say on whether it actually satisfies the user story.

Scrum Master

The Scrum Master organizes the daily scrums. If any issues arise which can't be addressed by the developers, he or she should facilitate resolution of the issues.

They will generally organize all the meetings required throughout the sprint.

The Scrum Master is not a manager in the traditional sense, but a facilitator. The development team is self-managing. However, the Scrum Master is responsible for understanding how much work the team can complete in a sprint. This ensures that most sprints successfully deliver their committed work.

Developer

Developers are responsible for creating and testing the code. The ideal is that all developers are interchangeable. There are not pieces of code that belong to someone.

There should be no more than 10 developers in a team. If more are needed, then separate scrum teams are created to work in parallel. Otherwise scrums become excessively long, and the team's productivity is hampered.

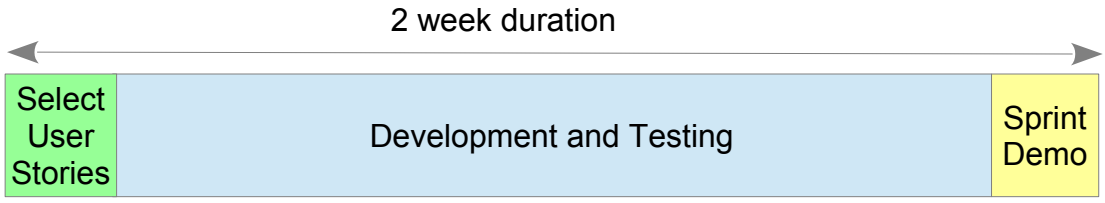
Process

The two key components of the Agile process are sprints, scrums, and backlog management.

Sprints

Sprints are typically of 2 week duration. On the first day of the sprint, the user stories to be delivered are selected. The Scrum Master ensures that the selected work can be delivered within a sprint.

The majority of the sprint is spent developing and testing those user stories. On the final day of the sprint, this work is demonstrated to the Product Owner.



During the Sprint Demo, the finished user stories are marked as complete and closed. If a user story was not completed, there are two possible approaches:

- If the user story is mostly complete, the remaining parts may be moved to a new user story to be scheduled for a subsequent sprint.
- If the user story is mostly incomplete, it is left open, and added to the next sprint.

Scrums

Scrums occur daily throughout the sprint, except on sprint demo day. Each developer describes:

- what they did yesterday
- what they will do today
- any obstacles that are impeding their progress

The scrum should take no longer than 1/2 hour. Topics outside the scope of the 3 items above should be held in breakout discussions after the scrum.

Backlog management

Requirements can be added to the backlog at any time, but there is a process for doing so. The entire development team meets with the Product Owner, and review each of the new user stories. They ensure that all user stories are well understood by the team, and contain all of the necessary information.

Since user stories must fit into a sprint, they cannot exceed a certain size. To ensure this the developers each estimate its size, and explain the basis for their estimate. If there is significant variation between team members, further discussion establishes a consensus. If a user story is too large to fit into a sprint, it is broken into smaller parts.

Conclusion

Agile development has become popular because it is extremely successful at delivering software on time. Changing requirements and design changes can be incorporated without breaking the process paradigm.

It also gives a lot of flexibility in how and when software is delivered. Since increments of functionality are developed frequently, and always completely tested, a new release of the software can be made at any time. When important customers request new features, they can be prioritized and delivered quickly.

Agile has evolved considerably since its initial invention, and what is described here is roughly the state of the art.