Working Agile with Scrum

Agile methodologies

History of Agile methodologies

Agile project methodology allows one to manage the complexity of today's business systems and make them more flexible. Agile system development takes into account requests for changes and reacts flexibly to new requirements that appear during the duration of the project. The method emerged in the 1990s but it wasn't until 2001 that agile went from an idea to a reality where 17 programmers were involved.

Agile methods have become increasingly popular in recent years, both in software development and other areas where speed and flexibility are important. Agile methods have proven to be effective for handling complex problems and for responding quickly to changing conditions.

Scrum

Scrum is an Agile software development framework that divides the software development process into sprints. The team works collaboratively in a cross-functional manner to deliver a product with incremental gains/improvements at the end of each sprint. The framework includes multiple meetings to ensure efficiency and emphasizes flexibility and adaptability to quickly respond to changes.

Kanban

Kanban is a Lean framework that visualizes workflows using a board with columns representing different stages of work. It limits work in progress and focuses on continuous improvement to maximize efficiency and minimize waste. It is often used in combination with other agile frameworks to improve software development.

Comparison of Scrum and Kanban

Scrum and Kanban are both agile frameworks for project management and software development, but they differ in a few ways. Scrum has a set time frame (sprint) and uses specific roles and routines to manage the project, including sprint planning, daily stand-up, sprint review and sprint retrospective. On the other hand, Kanban is more flexible and focuses on visualizing the workflow and managing constraints to deliver value continuously.

Kanban often uses a board with different phases for example "to do", "in progress" and "done" to show the workflow. The team then uses cues, such as cards or color-coded post-it notes, to mark when a task moves from one phase to another. The goal is to avoid bottlenecks and improve workflow by quickly identifying and handling potential problems.

Scrum and Kanban have their own advantages and are effective for different projects and teams. The choice between the two depends on factors such as the size of the team, the way of working and the degree of predictability required in the project.

Team’s experience with Scrum

Overview

In general, our team worked well with scrum. We held Daily Stand-ups, held meetings at the beginning of every sprint to plan and assign the different user stories to team members. We had sprint reviews at the end of each sprint where we discussed our progress with the customer. And we held retrospectives for the different sprints where we discussed challenges we faced when working with scrum and how to overcome them.

Some of the challenges we faced in the beginning was deciding on deadlines for the different user stories. We only set one deadline for all the user stories together instead of individual deadlines for them. It made it difficult as some developers had less time to work with their user story because some of them required the completion of others and could not be worked on in parallel. Another challenge we faced was that some daily stand-ups took longer than 15 minutes and sometimes turned into a discussion about the project. Some developers also failed to inform the scrum master about their absence from certain daily stand-ups, which lead to the delaying of said daily stand-ups. Another issue was not directly informing the absent members about changes made to the project as soon as possible, as they only found out about the changes at a much later date.

We discussed the challenges during the sprint retrospectives and managed to solve some of them during other sprints for example, we set deadlines while taking into account the different user stories and adjusted them accordingly. The scrum master also made sure to end the daily stand-ups after 15 minutes and any discussion that arose was interrupted and moved to a new meeting. We improved the communication between team members in general and had better cooperation with each other.

Our evaluation of the user stories was for the most part accurate. We only had an issue at the beginning of the first sprint, because we thought that the user stories were more demanding than they were which lead to us doing more than what was required.

We had four sprints in total sprints one to four. The scrum masters for the different sprints were Yosuf, Ilyas, Michael, and Ahmed, respectively. At the later sprints, some scrum masters also held the role of developers.

Requirements and user stories

During sprint plannings, we discussed the various requirements we had and created new user stories based on those requirements. Some requirements were turned into several user stories while the rest were turned into just one. The user stories were then assigned to the developers through taiga.

Sprint 1 had 8 user stories, user stories 1 to 8. The second sprint didn't have user stories as we did not receive any new requirements from the customer. Instead it had small task as improving some areas in our database which needed improvement, and writing the project documentation. The third sprint had 4 user stories, user stories 9 to 12. One of the user stories from sprint 3 was not completed so it was moved to sprint 4. For sprint 4, we only received one requirement from the customer which was to normalise the database to 5NF. We decided to turn it into a task along with the rest of the deliverables for the project.

Here is the link to the different sprints and their respective user stories:
https://tree.taiga.io/project/naviciin-pet-code-blasters/taskboard/sprint-4-1281