AGILE

S3-INDIVIDUAL

Welles,Nick N.H.M.

2022 - 2023

Table of Content

[Agile in my projects 2](#_Toc124508380)

[My use of Agile 2](#_Toc124508381)

[Scrum 2](#_Toc124508382)

[Story mapping 4](#_Toc124508383)

[User acceptance criteria 4](#_Toc124508384)

[Requirements / User stories 4](#_Toc124508385)

[What is Agile? 5](#_Toc124508386)

[Agile manifesto 5](#_Toc124508387)

[The 12 principles 5](#_Toc124508388)

[Scrum 5](#_Toc124508389)

[Sprint 6](#_Toc124508390)

[Agile requirements 6](#_Toc124508391)

[Functional requirements 6](#_Toc124508392)

[Non-functional requirements 6](#_Toc124508393)

[User stories 6](#_Toc124508394)

[Acceptance criteria 6](#_Toc124508395)

[User acceptance tests 7](#_Toc124508396)

# Agile in my projects

If you are reading this and you have no idea what Agile is, skip to page 5 or click the sentence below.  
[What is Agile?](#_What_is_Agile?)

## My use of Agile

From my understanding, [Agile](#_What_is_Agile?_1) is a method to easily cover parts, or sub-parts of a project and work them out. It is a method where you can use multiple tools to make the process easier for you, and your whole team. During semester 3 we made use of the Agile method. as in my individual project, we also used this method in the group project.

### Scrum

To apply the agile methodology for building my project, I used [SCRUM](#_Scrum). I was already familiar with this and so to continue with it right away was very convenient for me. You can find the board via the link below.

[View Scrum board of Instruweb](https://github.com/orgs/instruweb/projects/1/views/1)

[View Scrum board of WJJCN (group project)](https://github.com/orgs/wjjcn/projects/1)

As said before, we also made use of agile in the group project. Together with scrum. Every [sprint](#_Sprint) review (a sprint was 3 weeks) we held a review with our stakeholders (World of Content) and after that we held a retrospective. In that retrospective we discusses what we already did good, what we could add during the process, what we already did, but could do more and what we shouldn’t do anymore. After each retrospective we put those in [FeedPulse](https://fhict.instructure.com/courses/12512/external_tools/1067).

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Together with the retrospective, we all gave each other feedback on what we do good and/or what we could do better or should stop doing. This was also put in FeedPulse under the tab Peer feedback.  
We did the retrospective under the KALM (Keep, Add, Less, More) methodology. We put things in we did good, didn’t do and should do, should do less and should do more. This was very nice to keep track of and see what we did improve on compared to the previous sprint(s).

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Figure 1: Peer to peer feedback in FeedPulse

Afbeelding met tekst, whiteboard

Automatisch gegenereerde beschrijving

Figure 2: Picture of our retrospective after a sprint review

We also did sprint-poker. We all would give an estimate on a ticket/card in the scrum board, and somewhere where we were all satisfied with, we put the estimate in there.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Figure 3: Sprint poker in our Discord server

### Story mapping

To create a rough schedule of all the requirements, I also used story mapping. Here I made an overview per sprint of which requirement I want to have ready in that sprint.

The benefit of using the story mapping method with agile is that it helps you organize every requirement and sub-cards that that specific requirement has underneath them. If you are working in a team, the team will understand each post-it card better.

Story mapping can also be used with a process, for example when you would like to order a product. What steps are needed in order to complete that process?

[View Story map file of Instruweb](Instruweb%20-%20Story%20mapping.pdf)

### User acceptance criteria

We used this method in our group project, but more as the [user acceptance criteria](#_Acceptance_criteria). Every sprint review we made a planning for the next sprint which requirements or sub-cards we were going to work on. In consultation with WoC, we sometimes removed certain requirements and put others in their place. Every sprint, we would show what we had done about these requirements and which ones we were going to move on to the next sprint if we did not finish them in the same sprint.

[View planning per sprint](https://github.com/orgs/WJJCN/projects/1/views/7)

### Requirements / User stories

To define which features they wanted in their product, we discussed that the first two sprint reviews. After those sprints we had a clear vision of what we needed to setup as [requirements](#_Agile_requirements) to build the product.

[View old requirements file](https://github.com/WJJCN/Documentation/commit/369eb274d3fa4bd4200df1d6474a474346482169?short_path=f352eea#diff-f352eea7f6f774a1699fa919d20b39121b8e7d031f06559038f8a27212baff74)

So after two sprint reviews with WoC and feedback on our requirements and input on what they wanted in the product, we came with the final requirements document. These requirements where made in the form of [user stories](#_User_stories).

[View file](https://github.com/WJJCN/Documentation/blob/main/Documentatie/Requirements.md)

Research

# What is Agile?

In short, Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of setting all their sights on a resounding launch, an agile team delivers work in small, but immediately usable, chunks. Requirements, plans and results are continuously evaluated, giving teams a natural mechanism to respond quickly to change.

## Agile manifesto

In mid-2001, against the setting of the Wasatch Mountains in Snowbird, Utah, US, 17 individuals accumulated to examine the eventual fate of programming advancement. The issue, they concurred, was that organizations were so centred around unnecessarily arranging and recording their product advancement cycles that they were neglecting to focus on what truly made a difference: fulfilling their clients. Organizations might have promoted corporate qualities, for example, "greatness" and "trustworthiness," however these qualities did essentially nothing to assist with directing individuals in the correct heading, particularly programming designers. That expected to change. A large number of the Snowbird 17 previously had thoughts regarding how to introduce the new period of programming improvement. The outing to the mountains was their opportunity to examine this inside and out.

The consequence of this long end of the week was the Spry declaration, comprising of only 68 words. Notwithstanding, the short however exceptionally strong report changed programming improvement for eternity. In the almost twenty years since the record was made, these words (and the 12 rules that observed) have been embraced (to fluctuating degrees) by endless people, groups and organizations.

## The 12 principles

1. Early and Continuous Delivery of Valuable Software
2. Embrace Change
3. Frequent Delivery
4. Business and Developers Together
5. Motivated Individuals
6. Face-to-Face Conversation
7. Working Software
8. Sustainable Development
9. Technical Excellence
10. Simplicity
11. Self-Organizing Teams
12. Regular Reflection and Adjustment

Source: <https://www.plutora.com/blog/12-agile-principles>

## Scrum

So, Scrum is a system that assists groups with cooperating. Like a rugby crew preparing for a significant match, groups can utilize scrums to learn through encounters, sort out themselves as they deal with on an issue and ponder their victories and missteps to work on themselves constantly.

### Sprint

A sprint is a defined time period within which a scrum team completes a certain amount of work. Sprints are the core of scrum and agile working methods, and by executing sprints properly, your agile team can deliver better software with less effort.

## Agile requirements

Identifying agile requirements for a project can help teams understand what functions and features the finished product needs to have. It can also help set performance benchmarks so they can determine how well the product functions.

### Functional requirements

A functional requirement recognizes a capability or highlights the completed item needs to have. Groups utilize this data to figure out what steps they need to take to deliver the ideal item. They can likewise utilize this data to lay out objectives and benchmarks for creation to assist them with keeping tabs on their development. Example:

* A landing page with a customer feedback form
* A search feature that allows users to find past invoices
* A forum that members can use to communicate with each other

### Non-functional requirements

Non-functional requirement characterize how well an answer needs to perform. Otherwise called quality credits, non-functional requirements depict the overall qualities of an item or framework. They depict how it should act and lay out requirements on its usefulness.

Non-functional requirements are generally founded on something you can measure. Like convenience, security, unwavering quality and performance.

### User stories

User stories will assist with communicating prerequisites according to the viewpoint of an end client. Distinguishing user stories toward the start of an undertaking can assist with joining individuals survey which highlights are the most significant and foster items or administrations that address the issues of the individual getting them. They can likewise assist with joining individuals separate explicit item includes into more modest and more reasonable undertakings. Example:

* As a user, I would like to receive an email after registration so I can confirm my email address.
* As a customer service representative, I need to improve our customer response time so we can retain customers.
* As an investor, I need to see a daily summary of my investment accounts so I can focus on which one needs my immediate attention.

These user stories clearly define what features they may want and why. By identifying users' needs with user stories, teams can focus on producing higher-quality products and services.

### Acceptance criteria

The acceptance criteria characterize how a group can test or gauge a client story to decide it's working. Groups use acceptance criteria to recognize which measurements they can use to evaluate the progress of a task. Here are a few instances of acceptance criteria:

* Has customer retention improved by 15% over the last year?
* Is the speed of product dispatch under 24 hours?
* Has the product range increased by 20% over the last two years?

To develop acceptance criteria, make sure the requirements follow the SMART goal methodology. SMART goals are goals that are specific, measurable, achievable, relevant and time-based.

### User acceptance tests

User acceptance tests identify an actual set of scenarios a tester could go through to assess that a specific product or service feature is complete. This can help teams and clients understand how a feature works and verify if a solution meets the needs of their customers. For example, if a team wants to create a user acceptance test to make sure consumers receive a confirmation email after registering to download their software, they could implement the following user acceptance test:

* Step one: Visit the website www.softwaretest.com/register
* Step two: Register for the software by filling out the form with your email address.
* Step three: Open your email address and locate the confirmation email.

If a user completes all the steps in a user acceptance test, then the team can determine that the product meets the acceptance criteria.

Source: <https://www.indeed.com/career-advice/career-development/agile-requirements>