Bern University
of Applied Sciences

URL-Archiver

Intermediate Report

Course of study	Project 1
Author	Nicolin Dora, Abidin Vejseli & Kilian Wampfler
Advisor	Dr. Simon Kramer
Co-advisor	Frank Helbling

Version 1.0 of November 7, 2023

- School of Engineering and Computer Science
- Computer Science

Contents

List of Tables

List of Figures

1 Introduction

1.1 Initial Situation

The Internet is constantly evolving, which means that there is no guarantee that a website as it exists today will still exist in a few years' time, let alone contain the same information. While this might not be a concern that the average Internet user has to grapple with, it poses a challenge to the academic demographic, where it becomes crucial to reference sources and potentially integrate links to additional data. If links become inactive, verifying the sources becomes challenging, if not impracticable. Archiving the existing status of a website is achievable, but it currently necessitates a manual and hence time-intensive operation, which not many people take the time to do. The objective of this project is to devise an automated solution to this predicament that is independent of platforms. The stakeholders for this solution include:

- IT users possessing basic computer skills who can download and operate the program from Github
- Dr. Simon Kramer, the technical project supervisor and intellectual owner of the project idea

1.2 Poduct Goal

The product goal is a platform independent Java application called ”URL-Archiver”. The application must be Free/Libre and Open Source Software (FLOSS) licensed and fulfil the following functionalities:

1. The software should be CLI[1]-based and offer a clear command line.
2. The software should allow the user to input a path, which can be a folder or any Unicode text file.
3. The software examines the contents of a file or folder to extract any web URLs using a standard regular expression or similar method.
4. If desired, URLs can be automatically opened in a web browser.
5. The extracted URLs are archived on archive.today and/or web.archive.org as per the user’s preference.
6. The software outputs the resulting archive URLs to the user.
7. The software generates a CSV file containing the content URL & the archived URL.
8. Optionally, the archived URLs are stored in a .bib file.

The product goal is achieved if the software covers all the functionality listed above. Furthermore, the code should be minimalistic, modular, and self-explaining. In addition to the code, it is essential that the following documents are provided:

- User manual
- Installation instructions (including installation script)
- Software documentation

[1] Command Line Interface

2 Scrum Implementation

2.1 Allocation of roles

In this chapter, the Scrum roles (Product Owner, Scrum Master, Developer) and additional roles such as Customer, Stakeholder, etc. are defined.

2.2 Scrum roles

We have decided to structure our Scrum team in the following manner:

Role	Person
Product Owner	Nicolin Dora
Scrum Master	Abidin Vejseli
Developer	Nicolin Dora, Abidin Vejseli, Kilian Wampfler

Table 2.1: Scrum Roles

Nicolin took on the role of Product Owner as he had concrete ideas and visions for the product at the start of the project. Additionally, he took on this role because he wanted to deal with the subjects surrounding the product backlog.

Abidin took on the role of the Scrum Master as he has the most experience with the agile way of working. He has already had the opportunity to perform this role professionally on several smaller projects in the past.

Kilian took on the role of a Developer, as he is an active programmer in his job and has already gained some experience with Scrum. Therefore, self-organization is not a foreign concept to him.

Besides Kilian, all the other members of the group were also assigned the role of Developer, as otherwise the project would not have been feasible in the given time. This is due to the fact that we all work alongside the university.

2.3 Additional roles

In addition to the Scrum roles, we have assigned the following roles to our specialist lecturer and PM-coach.

Role	Person
Stakeholder	Dr. Simon Kramer
Customer	Dr. Simon Kramer
PM-Advisor	Frank Helbling

Table 2.2: Additional Scrum Roles

2.4 Sprint Goals

We have defined the goals of our past and current sprints in the best possible way according to the SMART[1] criteria. The goals of our sprints are listed below:

Sprint 1 Implement input handler for files (any unicode file e.g. .bib, .txt, .html and .pdf) and basic user guidance (Menu, Error messages).

Sprint 2 Implement a function to scan a provided text in order to identify and extract any URLs contained within it and upgrade our current console-based interface to enable users to easily open any extracted URL using their default web browser.

[1] Atlassian blogpost: "How to write smart goals"

2.5 Requirements

In this chapter, we present our product and sprint backlogs, structured according to Scrum.

2.5.1 Product Backlog

Our product backlog consists of user stories and epics created by our Product Owner. The user stories are prioritised and represent a set of initial requirements that must be met to achieve our product goal. The product backlog is maintained in Jira.

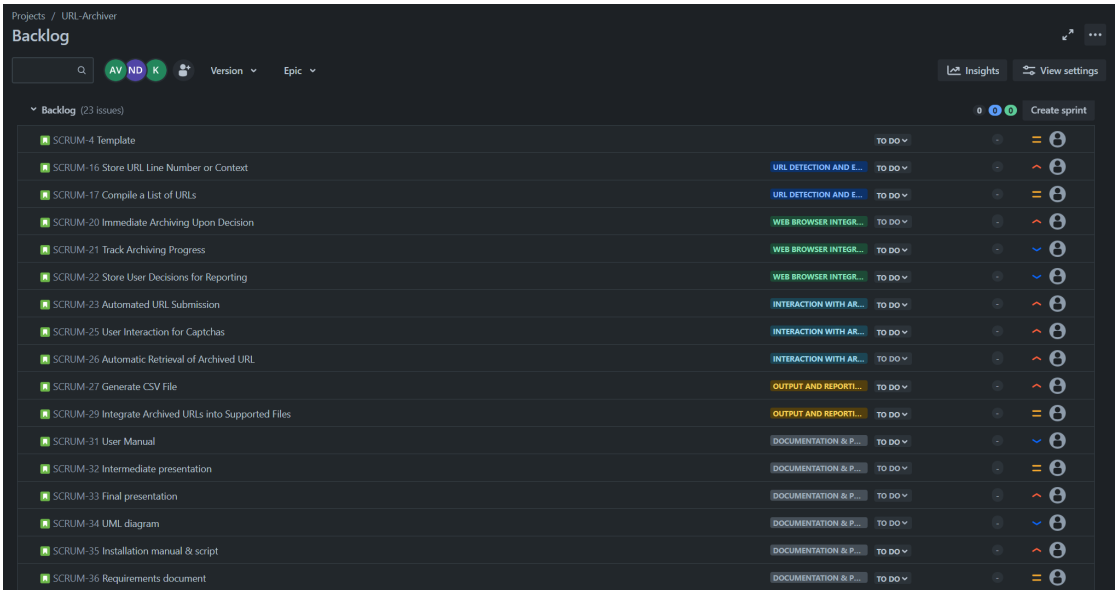

Figure 2.1: Product Backlog

The prioritisation of user stories in the backlog is based on the business value field, which has a value between one and ten. The business value is a vague estimate of how much value the individual user story has to the business, or in our case, to our stakeholders. We endeavour to estimate the business value based on the expected importance of the function to the stakeholder. The following priorities are possible:

- Highest
- High
- Medium
- Low
- Lowest

2.5.2 Sprint Backlogs

Below we describe our recent and current sprints. During the sprint planning we fill the respective sprint backlog with user stories that serve the sprint goal. A user story must satisfy our Definition of Ready before it can be included in the sprint. In addition, the stories must be estimated and the total number of story points must not exceed our defined velocity.

Sprint 1

Below is a screenshot of our board from the first sprint with the corresponding sprint goal.

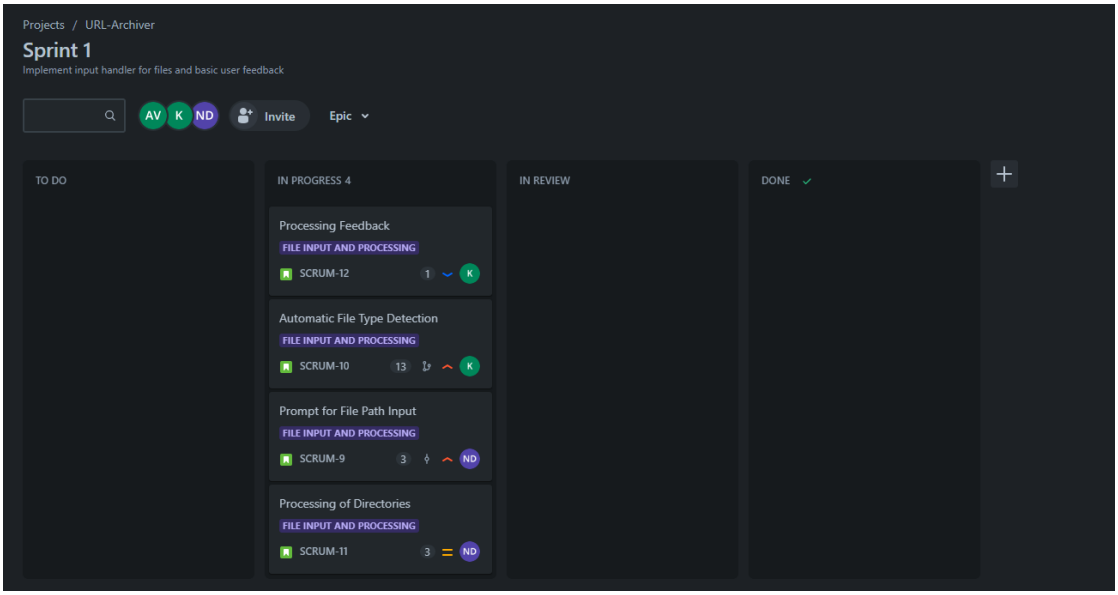

Figure 2.2: Sprint 1 Backlog

The user stories from the first sprint are shown below. The stories have been estimated and prioritised.

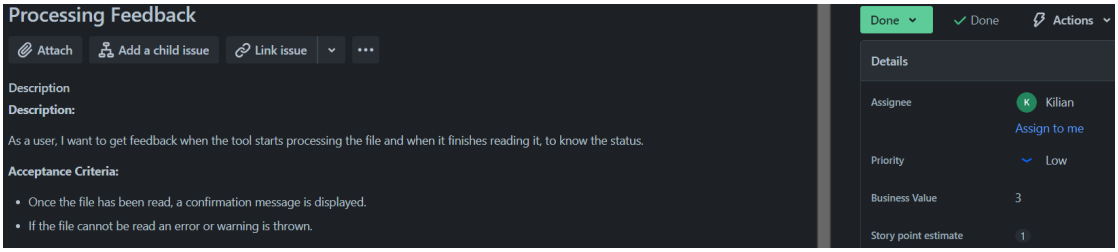

Figure 2.3: User Story Detail for "Processing Feedback"

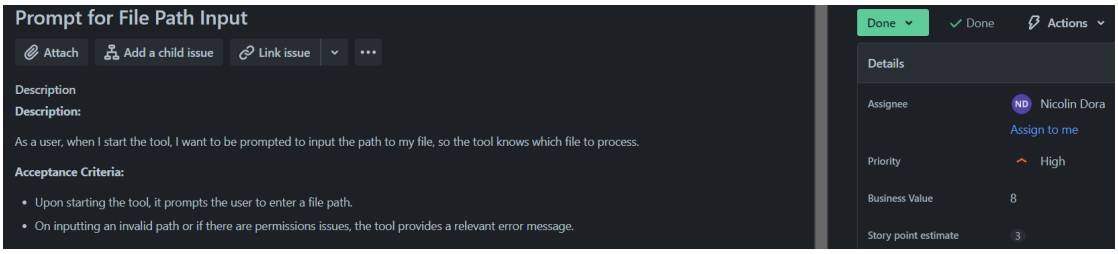

Figure 2.4: User Story Detail for "Prompt for file path input"

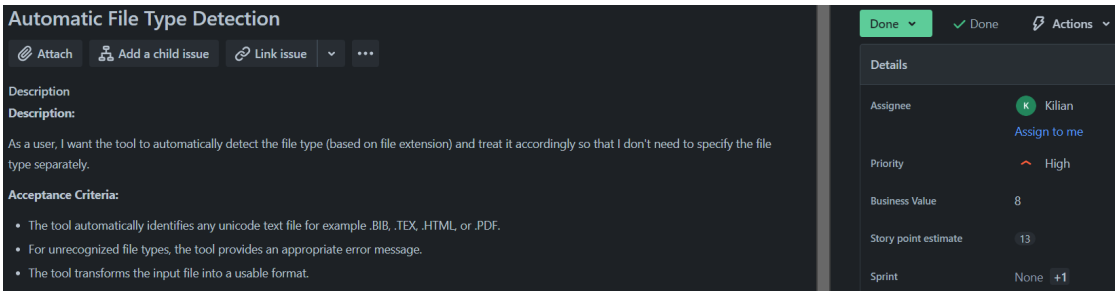

Figure 2.5: User Story Detail for "Automatic File Type Detection".

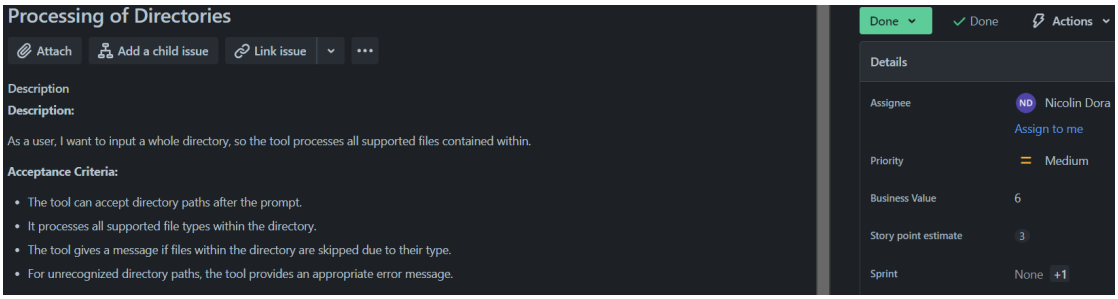

Figure 2.6: User Story Detail for "Processing of Directories".

In the first sprint, the burn up chart looks like this:

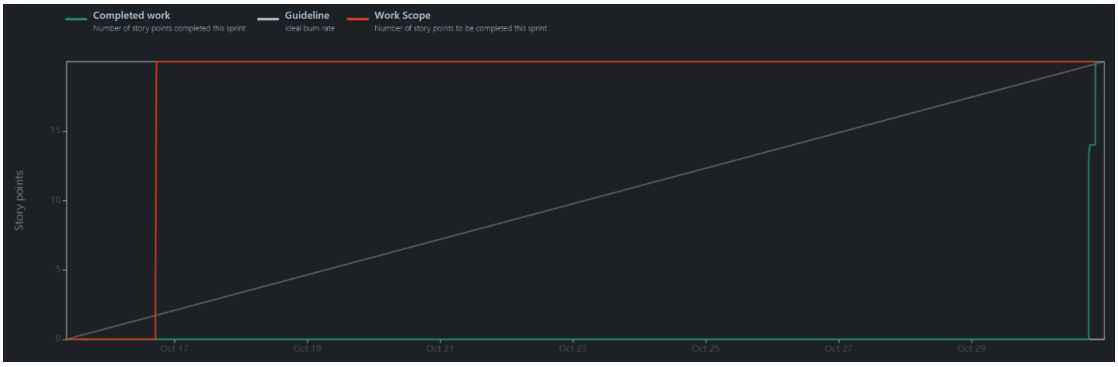

Figure 2.7: Sprint 1 Burn Up Chart

The reason for this is that we did not create tasks for our user stories as they were small enough. Furthermore, a code review for corresponding user stories could only be conducted towards the end of the sprint, resulting in the finalisation of user stories at that point.

Sprint 2

Below is a screenshot of our board from the second sprint with the corresponding sprint goal.

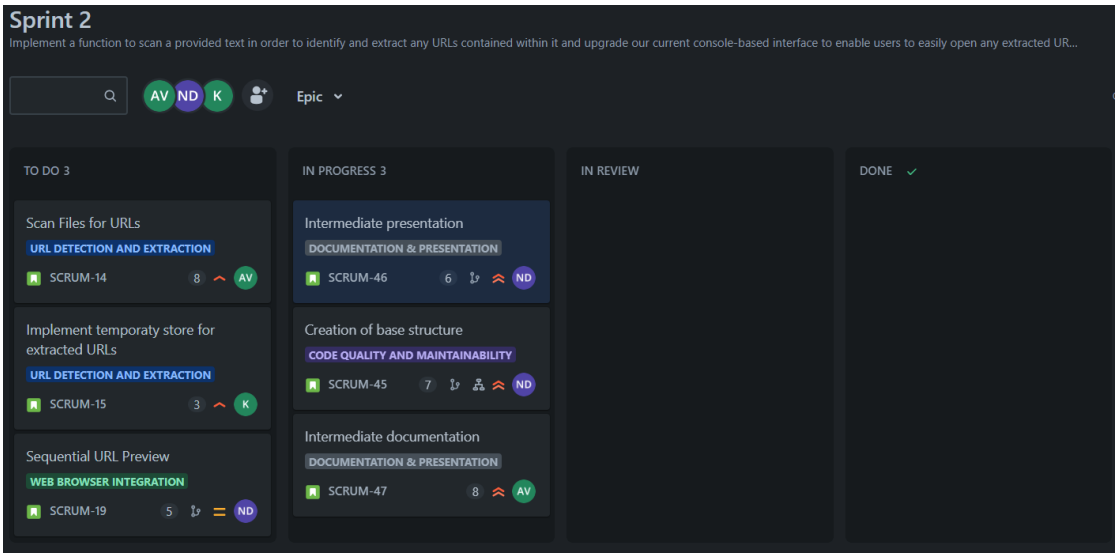

Figure 2.8: Sprint 2 Backlog

The user stories from the second sprint are shown below. The stories have also been estimated and prioritised.

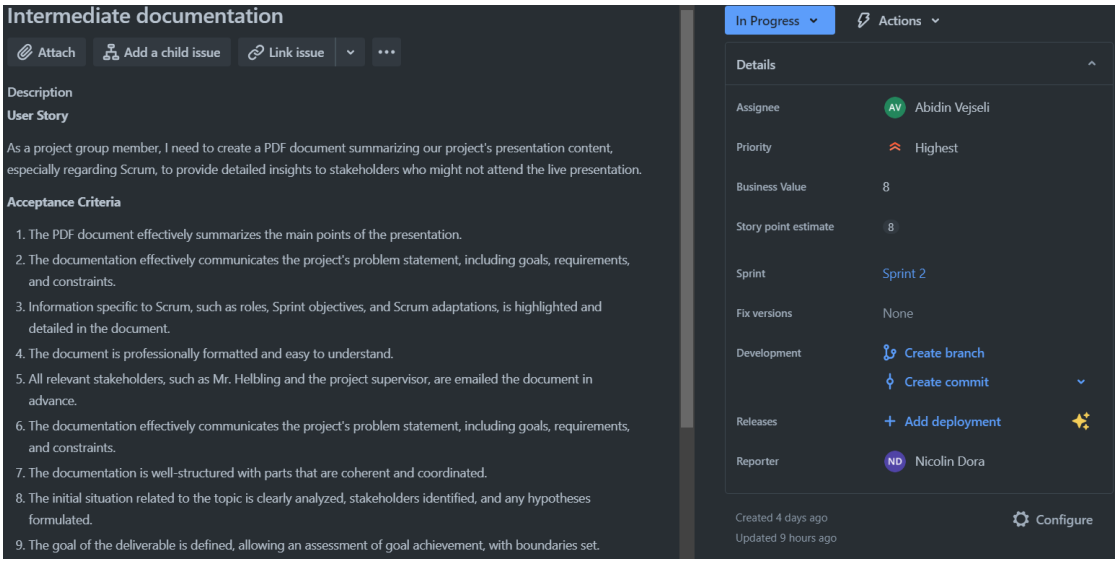

Figure 2.9: User Story Detail for "Intermediate Documentation".

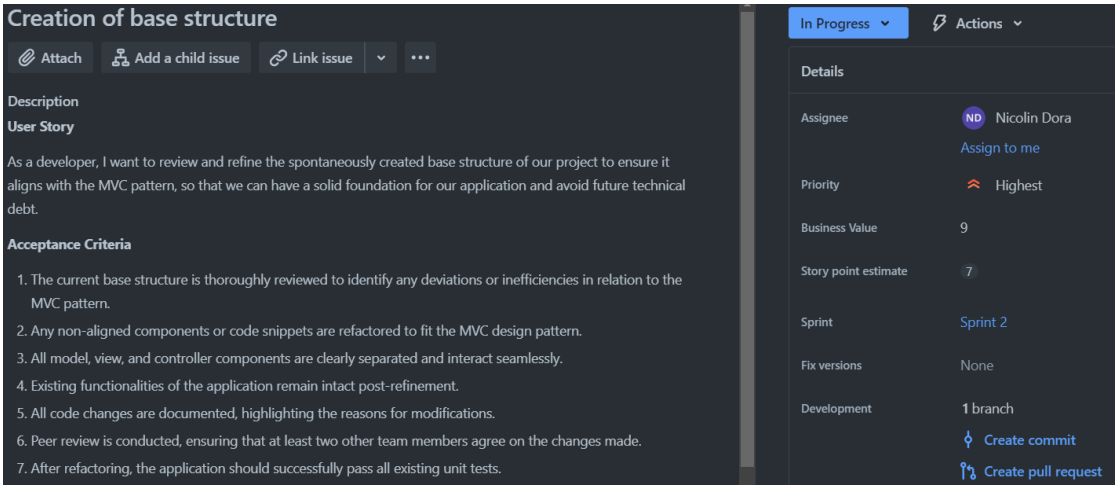

Figure 2.10: User Story Detail for "Creation of Base Structure".

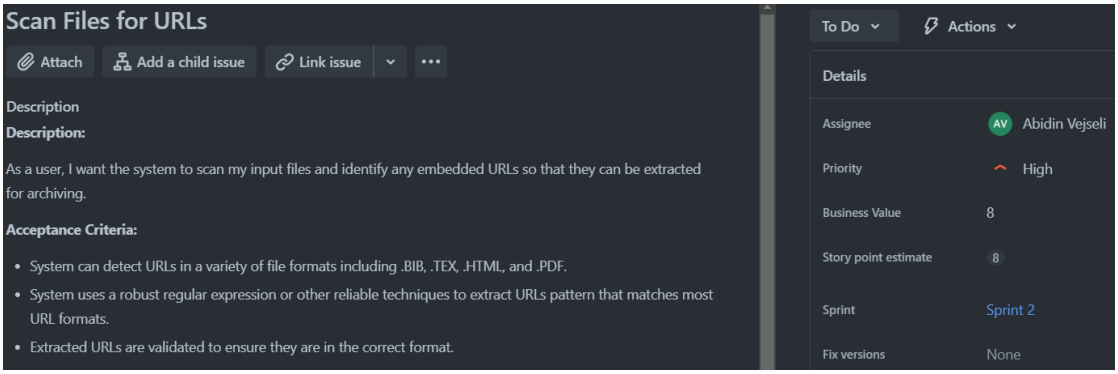

Figure 2.11: User Story Detail for "Scan Files for URLs".

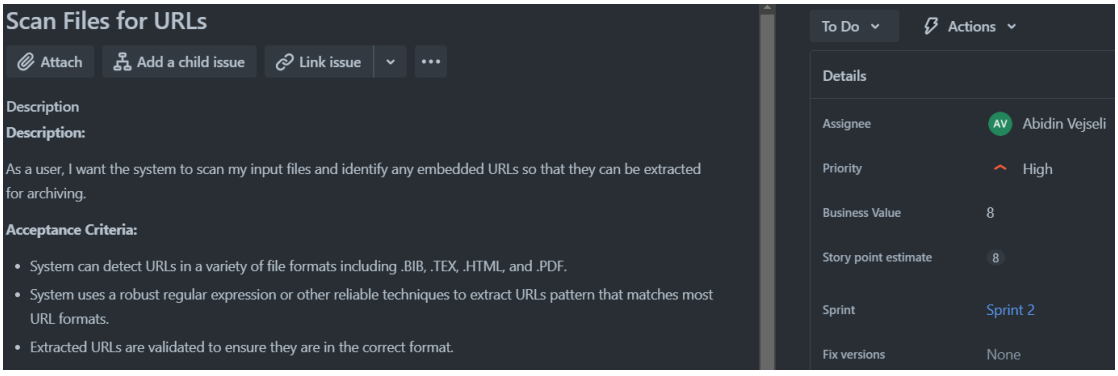

Figure 2.12: User Story Detail for "Sequential URL Preview".

In the second sprint, the burn up chart looks like this:

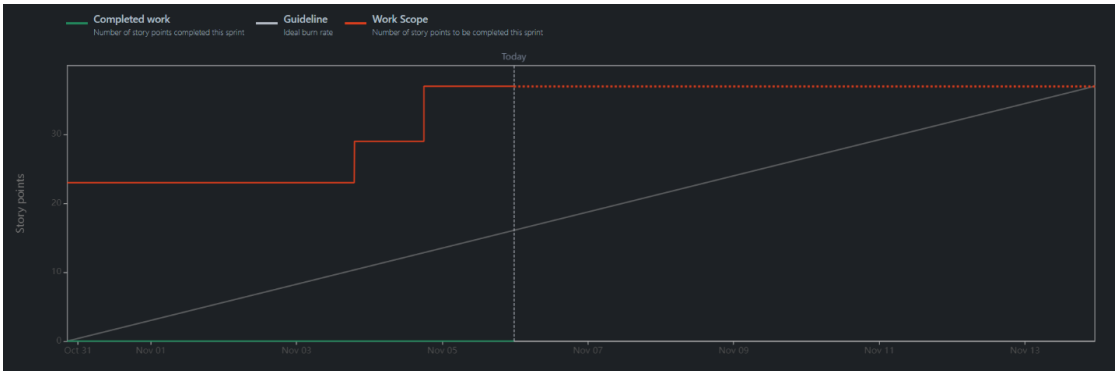

Figure 2.13: Sprint 2 Burn Up Chart

2.6 Scrum Adaptionen

As part of Project 1, we have adjusted Scrum in order to use it in the best possible way. The adjustments are explained in this chapter.

2.6.1 Definition of Ready (DOR)

Our DOR includes conditions that ensure that all team members understand the user stories and know when a user story can be included in a sprint. The DOR was set in line with the INVEST[2] criteria. A user story in the product backlog must meet the DOR before it can be included in a sprint.

Definition of Ready

- Ensure a clear definition
- Define the functionality or requirement to be implemented
- Clearly defined and testable acceptance criteria
- Ensure there are no or minimal dependencies
- Understood by the whole team
- The user story has been estimated
- The scope of the user story is small enough that it can be implemented in a single sprint.

[2] XP123 article: Invest in good stories and smart tasks

2.6.2 Definition of Done (DOD)

Our DOD contains all the characteristics and standards that a user story must meet to be considered complete. Once it satisfies the necessary quality requirements (acceptance criteria), the story can be considered complete and can be closed. The goal of our DOD is to create transparency so that everyone has a common understanding of when a story can be closed. A story that does not comply with the DOD may not be finalised.

Definition of Done

- Coding standards and best practices are implemented
- Unit tests for the feature are written and passe
- Any changes to the code or functionality are documented
- The code and functionality are reviewed by peers
- The feature works across multiple platforms
- Code is integrated with master branch
- Documentation has been updated
- Acceptance criteria are met

2.6.3 User Story Template

For the creation of a user story, we have defined a template so that the user stories contain all the necessary information. Below is a screenshot of our template. It includes all the relevant fields for us: Assignee, Priority, Business Value, Story Points estimate and assigned Sprint. Furthermore, we describe the user story in the ”Description” field in the format ”AS A ‹user role› I WANT TO ‹the goal› [SO THAT ‹reason›]” as well as the Acceptance Criteria. To ensure that we always have the DOR and DOD to hand, we also work with the Jira On-the-Fly add-on, which enables us to record both for each user story and tick off the individual points accordingly when they have been completed. This allows us to immediately recognise whether a user story can be included in a sprint and whether a story has been fully completed.

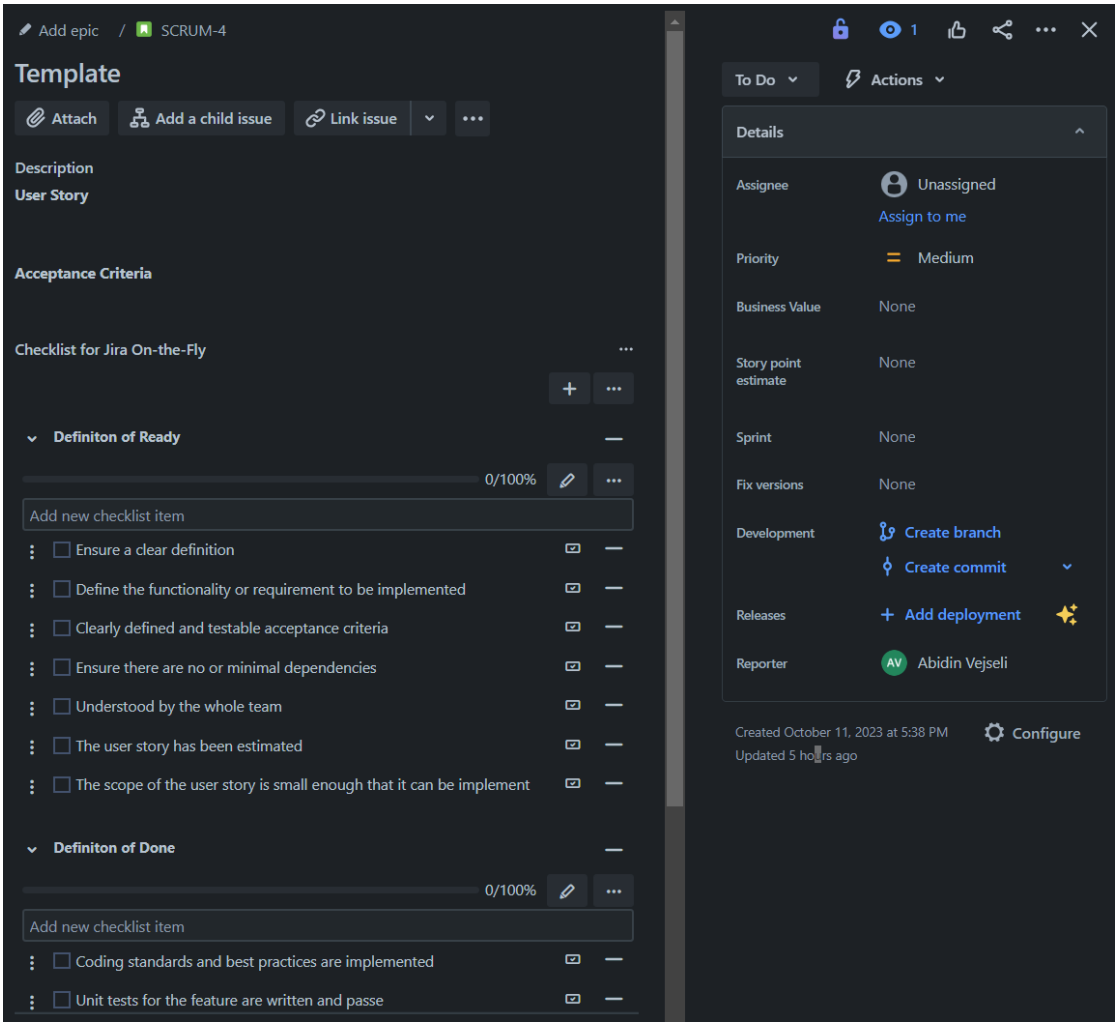

Figure 2.14: Screenshot from the user story template.

2.6.4 Estimation method

We have chosen the ”T-shirt sizes” method because it is a simple way to estimate effort (story points). This method is based on the fact that everyone knows T-shirt sizes and that large sizes mean more work than small sizes. As a result, this method enables us to make efficient estimations, despite the lack of shared experience in the team.

Below is the scale we use:

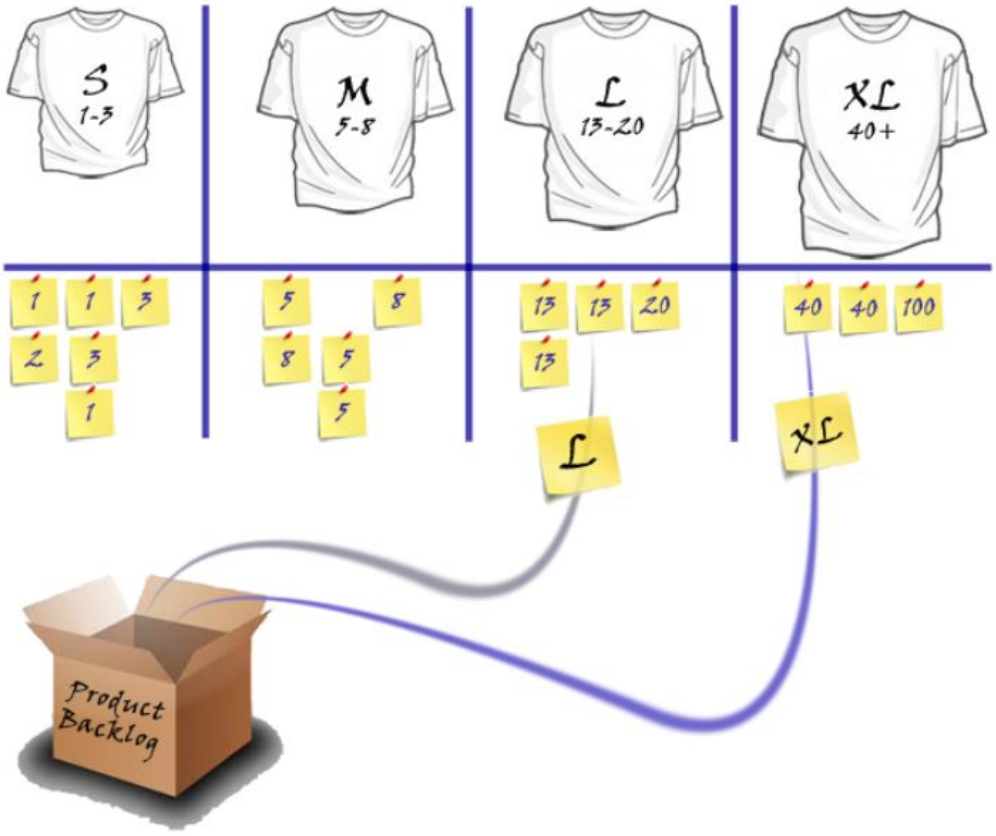

Figure 2.15: Ilustration of T-Shirt Sizes

2.6.5 Velocity

To select the user stories and tasks to be worked on, a suitable criterion, velocity, is applied to estimate what can be completed in the upcoming sprint.

We have decided that we have a velocity of 30 story points per sprint. Therefore, our workload per sprint should not exceed this threshold. We consciously take this into account during sprint planning.

2.6.6 Sprint

As a team, we have decided that our sprints will take place at two-week intervals. For each sprint we define a SMART sprint goal, which specifies the relevant user stories.

We decided in favour of the two-week rhythm because regular feedback is important to us and thus creates a greater learning effect. Additionally, we ascertained that one-week sprints would result in excessive overheads due to the administrative work involved in Scrum. Likewise, we consider sprints longer than two weeks to be impractical, as the interaction would suffer.

Sprint Planning

As part of sprint planning, we make decisions about which user stories can be implemented based on the sprint goal, the story points and the velocity. Before a user story can be included in the sprint, it must be estimated by the Scrum team. This task is always carried out at the start of our sprint planning. A sprint goal is then defined based on the estimated and prioritised user stories. The user stories we select for the sprint are based on the business value, priority, story points and velocity of the team. The sprint planning takes place on the first Monday of each sprint. We have decided not to have a second sprint planning as we have already defined in our DOR that the user stories should be as small as possible. In addition, each developer has the opportunity to divide their user stories into tasks within the sprint. This allows a better overview of the progress in the sprint.

Daily Scrum

As a team, we have decided not to have daily Scrum meetings, as this is not possible because all team members do work part times. Instead, we have two weekly meetings (weeklys), on Wednesday and Friday at 17:00, which last a maximum of 15 minutes. In addition, we have chosen to hold the meetings through Microsoft Teams as it is easier to organise. The goal of these meetings is to share the current progress, address issues, update the team and briefly discuss the next steps.

Sprint Review

In the sprint review, we check the intermediate result of the processed user stories. We check whether all the stories that should be completed meet the DOD. Furthermore, we discuss in the team what went well, what problems we encountered and how we solved them. Based on these results, the product increment is created. In our case, the product owner, who represents the customer, tests the product increment against the requirements. The review is done from the customer's perspective by testing the product increment. The outcomes are utilized to update the product backlog. The sprint review meeting takes place on the last day of the sprint.

Sprint Retrospective

In the sprint retrospective, we gather information as a team about what went well and what didn't go as planned in the previous sprint. We then derive specific improvements and plan their implementation. Our goal is to improve the efficiency, quality, communication and speed within our team. To achieve this, we give ourselves constructive criticism and are open to feedback. The sprint retrospective takes place on the last day of the sprint.