Scrum Document

V1.0.0

Author: Mehdi Haghshenas

September, 2021

Table of Contents

1. Scrum Definition

Scrum is a framework that helps people, teams, and organizations generate value through adaptive solutions for complex problems.

Our product development process RIGOUROUSLY follows the Scrum framework. It is expected that every member of the team trains, learns, and follows this framework and contributes to its evolution within the organization.

Scrum framework and its different events will be introduced in this document.

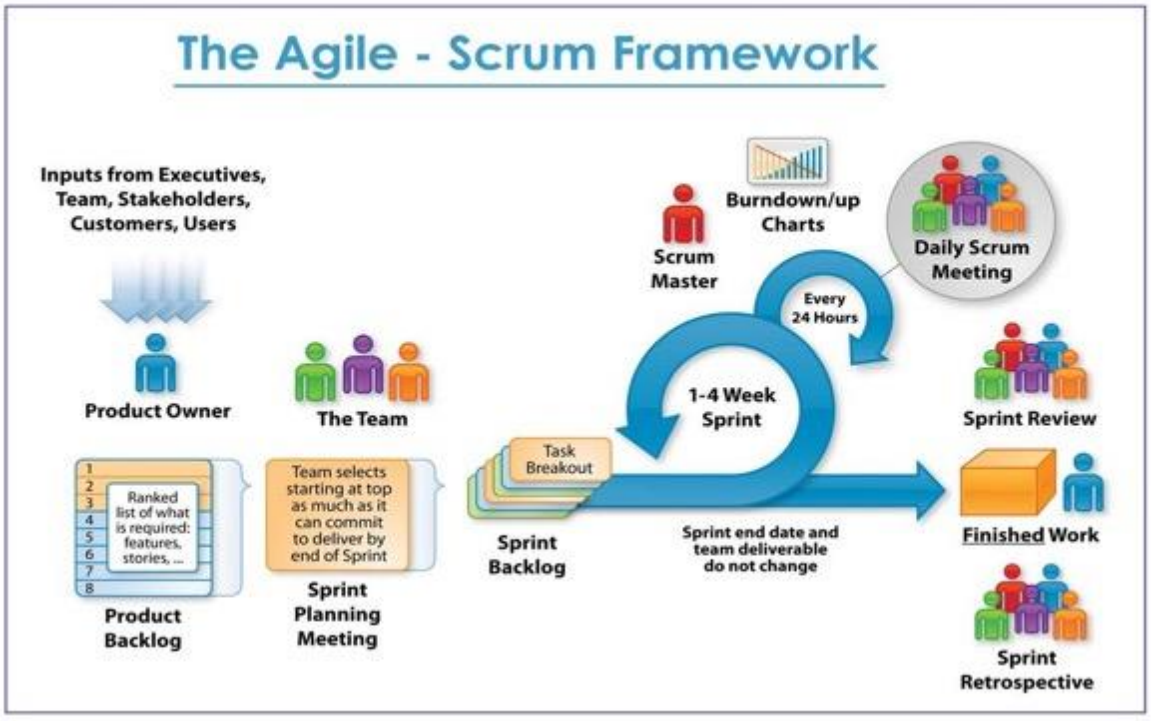

Figure 1. Scrum activities

2. Scrum Roles

The following are the roles in Scrum:

1. Product Owner
2. Scrum Master

3. Team

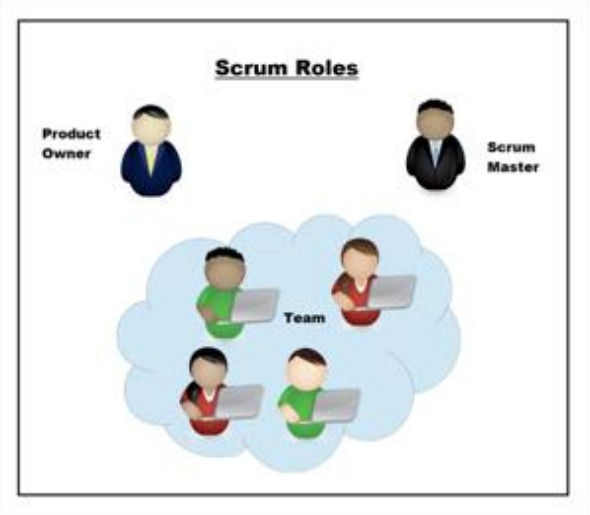

Figure 2- Scrum Roles

3. Scrum Structure

Scrum framework has a hierarchy structure as is depicted in figure 3 below. Each field is elaborated in more details.

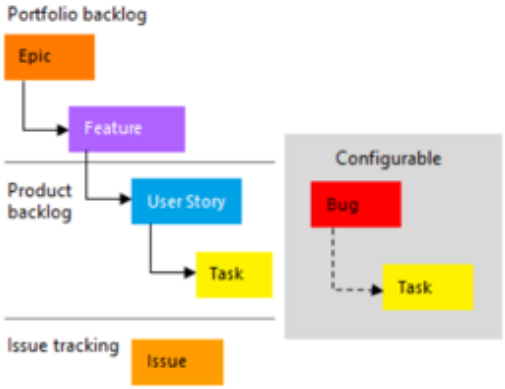

Figure 3. Hierarchy structure of the scrum framework

3.1. Epic

An **Epic** is a large body of work that is required to deliver one key aspect of the product. Epic can be broken down into several **Features.** Epic has a dynamic

scope. As the team learns more about an Epic through development and customer feedback, features or user stories in an Epic may be added and removed as necessary. Epic does not contain all the details that the team needs to work on. These details are defined in User Stories. An Epic usually takes more than one sprint to complete.

We usually use **Epics** as items that will be fully delivered in **months**, **features** as items that can be delivered in **weeks**, **user stories** as items that can be delivered in **days,** and **Tasks** as items that can be delivered in **hours.**

You can see an example for Scrum structure for a real product below.

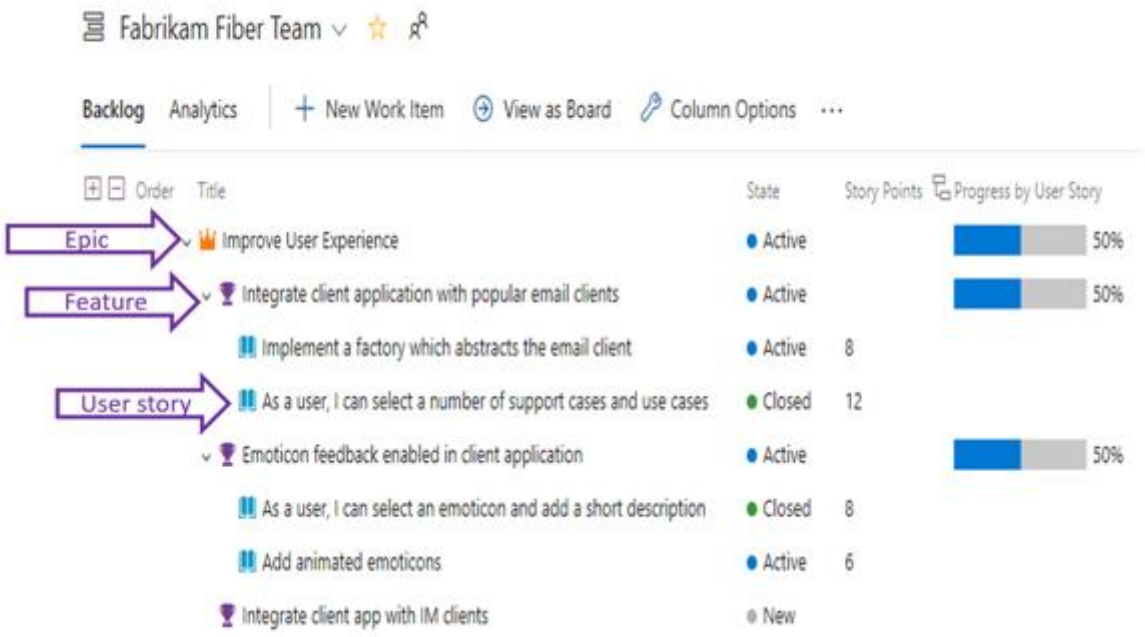

Figure 4. Demonstration of an Epic, a Feature, and a User story in the Azure DevOps

3.2. Feature

The **features** represent a chunk of functionality that delivers considerable business value and fulfills a stakeholder need. Each feature includes a benefit hypothesis and acceptance criteria. Features are a collection of user stories.

In our process, the Product Owner owns the Features. That doesn't mean that (s)he writes them, but has the final say on the content and prioritization of the feature.

The **Epic Sheet** is document we use that lists the product Epics and Features and has a roadmap to show the priority of features for the team

For each feature, we have a **Design Doc,** that includes the business value and description of the feature. The Epic Sheet and Design Doc are references for the scrum master and the team to break down the Feature into User Stories and their acceptance criteria and Milestones. The Design Doc also includes the technical design overview for each feature.

Figure 5. Feature detail in the Azure DevOps

3.3. User Story

Each feature is divided into multiple user stories. User story is a general explanation of a software feature written **from the perspective of the end user or customer**. User stories are a few sentences in simple language that outline the desired outcome. They don't go into detail.

The purpose of a user story is to articulate to the team how a piece of work will deliver a particular value back to the customer.

Writing user stories has the following advantages:

- **Stories keep the focus on the user.** A collection of stories keeps the team focused on solving problems for real users.

- **Stories enable collaboration.** With the end goal defined, the team can work together to decide how best to serve the user and meet that goal.
- **Stories drive creative solutions.** Stories encourage the team to think critically and creatively about how to best solve for an end goal.
- **Stories create momentum.** With each passing story the development team enjoys a small challenge and a small win, driving momentum.

Design Doc of a feature will include various use cases of a customer and the priority for product release. The team will then analyze the use cases and create a list of User Stories and the **Acceptance Criteria** for each User Story.

3.3.1. Acceptance Criteria

The **acceptance criteria** (AC) can be used as the basis for acceptance tests so that team members can more effectively evaluate whether a User Story has been satisfactorily completed. Before starting the **Grooming** session, Acceptance Criteria must be written as clearly as possible for every story. Conversations between the team and customers to define the acceptance criteria will help ensure that the team understands the customers' expectations.

What is written in the AC, technical requirements, or business requirements?

Only the **business requirement** is written in the AC. **Technical requirement** will be partly summarized in the Design Doc and discussed by the team members in the **design review** sessions for each story.

The below figure is a sample of writing AC for a user story. The below AC is written from the perspective an actual User for the User Story: " System user signs in with valid credentials"

Figure 6. A sample of writing AC for a user story

- **What are Acceptance Criteria Used For?**
  - ✓ **To define boundaries.** Acceptance criteria help development teams define the boundaries of a user story. In other words, acceptance criteria help you confirm when the application functions as desired, meaning that a user story is completed.
  - ✓ **To reach consensus.** Having acceptance criteria synchronizes the development team with the client. The team knows exactly what conditions should be met, just as the client knows what to expect from the app.
  - ✓ **To allow for accurate planning and estimation.** Acceptance criteria scenarios allow for the correct division of user stories into tasks, so user stories are correctly estimated and planned.

Each User Story can have various status during the software development process. The flow of a User Story and its status is shown in figure 7.

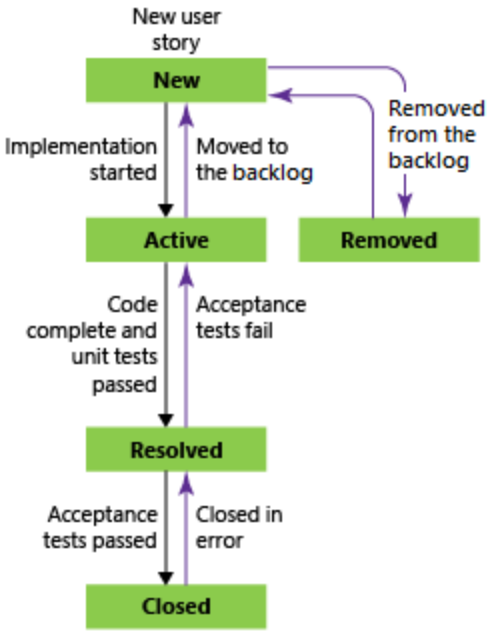

Figure 7. The flow of a user story and its status

- At the first step, a User Story in the **New** state.
- The team updates the status to **Active** when they decide to bring the User Story from Backlog to the sprint and complete the work during the sprint.
- A user story is moved to **Resolved** when the team has completed all its associated tasks and unit tests for the story pass
- A user story is moved to the **Closed** state when the product owner agrees that the story has been implemented according to the Acceptance Criteria and acceptance tests pass

In the Azure DevOps, each user story has different items which is depicted in the figure.7.

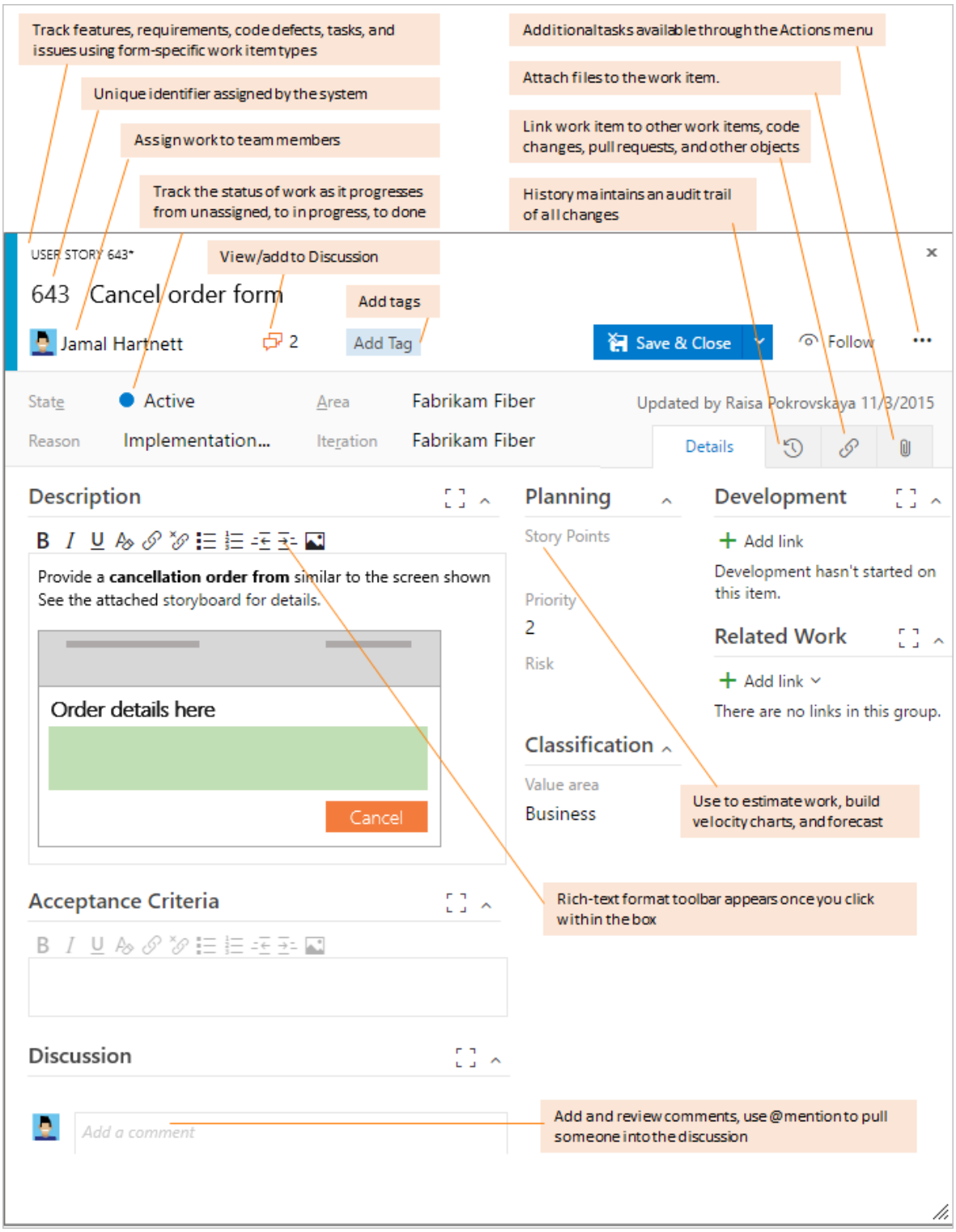

Figure 8. Introducing different items of the user story in the Azure DevOps

3.3.2. Definition of Done (DoD)

A User Story must meet the conditions defined in the DoD so it can be considered completed and closed. These conditions are pre-defined guidelines or standards shown in Figure 9 below.

In summary, the team member who is assigned the User Story is responsible for Dev Validation steps. The PR Review step includes going over the technical checklist, such as code review done, technical design validated, functional tests are validated. After the PR review completion, the Dev team hands the User Story to the QM team that is responsible for business requirement testing and verifying that ACs are met. The product owner or its representative is responsible for the Business Review step.

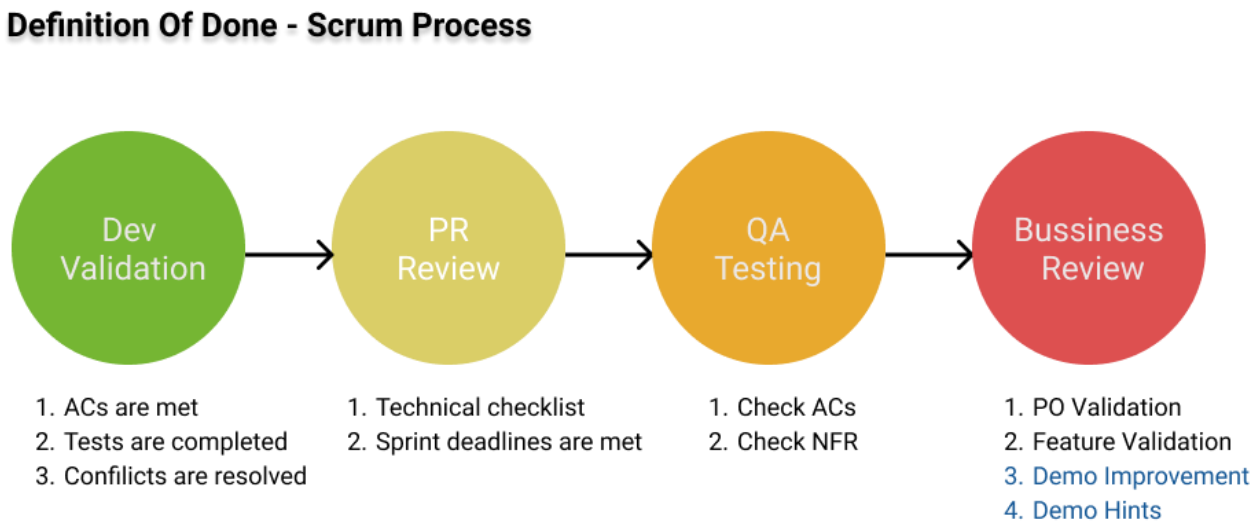

Figure 9. A sample of DoD items

3.3.3. Story Points

Story point estimates the amount of work required to complete a user story using any numeric unit of measurement your team prefers.

We use Fibonacci sequence (1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.) for story point estimates. The below table defines an estimated time equivalent for each story point.

Table 1. Estimated Hour for each story point scale

Story point	Hour per points
1 pt	1-3 hr
2 pt	3-5 hr
3 pt	5-8 hr
5 pt	9-15 hr
8 pt	16-24 hr
13 pt	25-44 hr
21 pt	45-80 hr

How can we assign a point to the story?

When calculating story points, you're looking at the total effort involved in completing the DoD and completing the required functionality in the User Story AC. In order to estimate the user story properly, team will need to discuss questions like:

- How complex is the work?
- How much work is needed?
- What are the technical abilities of the team?
- What are the risks?
- What parts are we unsure about?
- What do we need in place before we can start or finish?
- What could go wrong?

Important Notes:

1. If team is having trouble estimating a story point or the scope of work is overwhelming, the story MUST be broken down into smaller parts and split across multiple stories.

2. After some time working together most teams will have a good idea about how much effort is involved in each story point.
3. During story point estimation, all required effort for **Dev Validation, PR Review, and QM** steps of the DoD must be considered.
4. We strongly prefer to break down User Stories such that largest story point per each user story is 8 points.
5. If we realize during the sprint a story is over or underestimated it is OK to change the story points according to the new insights, **learn from it** and do a **better estimation** in the next sprint.

3.4. Task

Task is the smallest unit of work in the Scrum framework. Each User Story is broken down into multiple Tasks. Tasks are assigned during Sprint Planning.

the Scrum Master monitors and advises team to follow the process. During each sprint, the progress of the tasks is monitored by the spring task board. The task board is

3.4.1. Team Task Statuses – Task Board Columns

each Task can have one of 3 statuses: New, Active, Done.

each Bug can have one of 4 statuses: New, In Progress, Resolved, Done.

Each status has an owner. The owner of the status is responsible to complete the requirement then update the status and update the task in task board

New Status

- Status Owner: Developer
- When task is not started yet, or
- When task is blocked because it has a dependency or pending backlog, thus developer cannot continue his/her task (add a '**pending tag**' as shown in figure.10.)

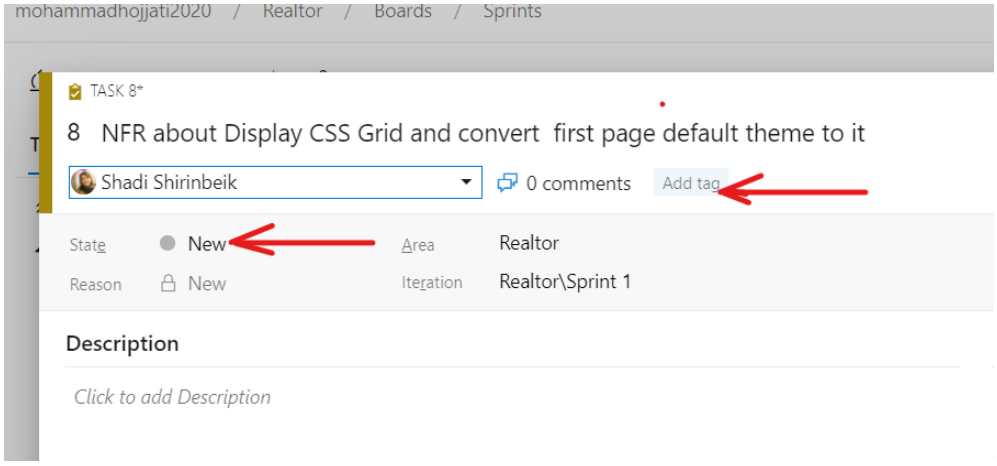

Figure 10. A sample of Task with a backlog Tag.

In progress Status (Active)

- Status Owner: Developer
- When developer is working on the task for development
- When task is implemented by the developer and he/she is testing it. In this status, each developer should use **'personal deployment'** to complete all tests, validate there are no bugs, and completed the Dev Validation step of DoD.

PR Review Status (Resolved)

- Status Owner: Tech Lead
- When task is completely done and tested, the developer moves task into PR review column. At this step, reviewers (tech leader) should check if task is done correctly without any bugs. Leaders should use '**personal deployment**' for testing as well.

Notes:

- **Maximum** deadline for each leader and reviewers to review the PR is one working day.

Done

- After Demo session, when product owners observe the outputs, if their expectations are met completely, and they are satisfied with the output, the task status will be closed.

3.5. Priority

Priority is a subjective rating of the user story, a feature, or a requirement as it relates to the business.

Allowed values are:

Priority = 1: Product cannot ship without the feature.

Priority = 2: Product cannot ship without the feature, but it doesn't have to be addressed immediately.

Priority = 3: Implementation of the feature is optional based on resources, time, and risk.

3.6. Risk

A subjective rating of the relative **uncertainty** around the successful completion of a user story. Allowed values are:

Risk = 1 - High

Risk = 2 - Medium

Risk = 3 – Low

3.7. Design Review Task

Each user story can have a design review task. Lead and Developer assigned to the story participate and discuss technical requirements and AC. If the User Story is based on the Design Doc, the owner of the Design Doc should record the design review in the Design Doc. If there is no Design Doc, both **lead** and **developer** should add a note to the User Story to summarize their design review session and encapsulate which technical solution is suggested for the user story.

4. Transparency and communication

Transparency is vital to the Scrum process, as it allows everyone to see and understand what is really happening in each sprint, achieving a bigger and better communication and

trust on the team. Without **full transparency** and **clear and timely communication** among team members many bad things can happen, including:

- Decreasing the team focus on what needs to be delivered
- Team morale can suffer
- Measuring future work is more difficult
- The team's true velocity is not known
- Lack of trust with the Product Owner

we use Azure DevOps to run the scrum process. All Communication MUST BE WRITTEN and can happen through the discussion section of the Azure DevOps. We can also use Teams channels with link to Azure DevOps User Stories and Tasks.

5. Scrum Ceremonies:

we follow the Scrum events and ceremonies carefully. Attending at these events is necessary and all the involved members should contribute at these meetings.

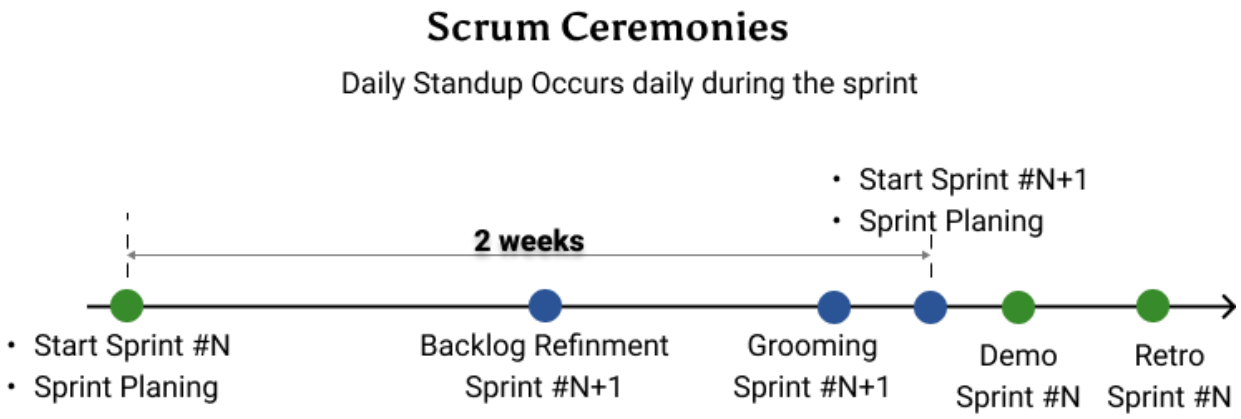

Figure 11. Scrum ceremonies Timeline

5.1. Backlog Refinement

In Scrum, Backlog Refinement is an ongoing process in which the Product Owners, Scrum members and the Development Leaders collaborate to ensure which functionalities and features should be delivered in the next sprint.

Dev. Tech Leads and Scrum team must estimate the progress of the current sprint and carryovers before this meeting. In addition, Epic Sheet and Design Docs are typically reviewed and a Backlog Refinement document is created. The team has one week to use the reference document to break down into user stories, identify dependencies, align with Design Doc milestones, and add ACs and prepare for Grooming.

Meeting Time: first week of current sprint on Wednesday (7:30 am) or tuseday (7:30 PM)
Duration: 1hr max

5.2. Grooming session

User stories will be explained to the team members by leaders, and after point poker, the story point is assigned to each user stories. All team members should join to this session. AC of each user story must be determined and completed before starting the grooming session.

Grooming session will be held before starting the Sprint on Thursday (7:30 am to 9:00 am).
Duration: 2hr max

5.3. Sprint Planning

In this ceremony, user story will be assigned to the developers.

It will be held on the first day of a sprint on Sunday (8:30am).
Duration: 1hr max

5.4. Daily (Stand up)

This ceremony is held every morning during a sprint. The intent behind a daily stand-up is that the team come together for a status check – to make sure that everyone is aligned and has visibility over what is going on, good and bad.

Time: (8:30am to 9am)

Duration: 15min to 30min.

5.5. Demo session

At the end of each sprint, Team members, either scrum team or the owner of a major User Story will demo the user stories which have been completed and met the DoD before the Code Freeze.

5.6. Retrospective session

This session will be held once or twice a month, the day after Demo meeting.

The Sprint Retrospective concludes the Sprint. It is timeboxed to a maximum of 1.5 hours for our 2-week Sprints.

During the Sprint Retrospective, the team discusses:

- What went well in the Sprint
- What could be improved
- What will we commit to improve in the next Sprint

Several issues that can make the team far from progress and prosperity will be discussed in the Retro session. An action item (AI) list will be created and the Scrum team is responsible to make sure the AIs are completed.

6. Scrum Process Summary (Cheat Sheet)

6.1. Key Reference Documents

Document	Owner	Includes	Comments
Scrum Process (this doc)	Scrum Master	Scrum Process	As we grow and evolve the process this document will be updated
EPIC Sheet	PO and Head of Engineering	EPIC and Feature Roadmap	Incudes references/links to Design Docs
Feature Design Document	Assigned Developer	Technical Design, Milestones, User Stories	Team can edit/comment PO may initiate this doc to add feature description, use cases and feature AC
Backlog Refinement Features List	PO	List of priority features for upcoming sprint	Discussed and written in each Backlog refinement meeting
Acceptance Criteria	Scrum Master or Tech Lead	The AC for each User Story written in the Azure DevOps environment	Must be completed before every grooming for all User Stories of the next sprint

6.2. Task Status and Task Board Columns

It is used to make sure every User Story and Task is making progress in the sprint and is following the required steps for Definition of Done (DoD)

Task Status / Task Board Column	Owner	Owner Responsibility
New	Assigned Developer	Tag if pending dependency
Active	Assigned Developer	Implement Complete Tests & Dev Validation Use Personal Deployment
Resolved	Check PRs Assigned Tech Lead	Technical Checklist, Code Review, Tests, Merge Complete Review within 24 hours
QA Review	QA Team member	Validate AC, Functional/Nonfunctional tests Report Bug if any
Done	PO	Complete After Demo Report Bug If Any

6.3. Important Guidelines

- Always use written communication in Azure DevOps or Teams. Tag appropriate team members.
- Tech Leads are responsible to break down features to User Stories for next sprint during the week between Backlog refinement and Grooming
- Scrum Team is responsible to make sure ALL User Stories have complete AC prior to grooming
- It is strongly preferred to keep estimated User Story points at 8point max
- There may not be more than 2 tasks IN PROGRESS for each team member
- Each team member is responsible to regularly (daily) update the Remaining Effort for active tasks

6.4. Frequently Asked Questions

1. **How should the AC be written, based on technical requirements, or business requirements?**

   Only the **business requirement** is written in the AC. **Technical requirement** will be partly summarized in the Design Doc and discussed by the team members in the **design review** sessions for each story.

2. **Are we allowed to add task / user story in the middle of the sprint?**

   If the **scope** of a user story does not change, then adding a new **Task** during the Sprint is **acceptable**.

   It is strongly preferred NOT to add any new User Story in the middle of the Sprint. Team must discuss about it in the **Retro** session to see why adding Story was needed. In rare cases, if it is absolutely required to add a user story at the middle of the sprint, another current user story should move to the next sprint and instead a new one added.

   Adding a new story should be always coordinated with the Crum Master and the team has the option to decline adding the new story in the middle of the sprint. It cannot be forced on the team.

   New scope should always be a **new story and should follow above guideline for adding to the sprint**.

3. **Are we allowed to change story points in the middle of sprint?**

   Typically, not. However, if we realize that we have overestimated or underestimated the points when the tasks start, it is usually because the Story was not properly analyzed by the Tech Leads analysis and not discussed or understood well in the Grooming sessions. In this case, it is OK to change the story points according to the new insights, **learn from it** and do a **better estimation** in the next sprint.

Reference:

1. https://www.pluralsight.com/courses/introducing-scrum
2. https://www.pluralsight.com/courses/real-world-scrum-team-foundation-server-2013
3. https://www.pluralsight.com/courses/agile-team-practice-fundamentals
4. https://www.oreilly.com/library/view/writing-user-stories/9781491993934/
5. https://docs.microsoft.com/en-us/azure/devops/boards/work-items/guidance/agile-process-workflow?view=azure-devops
6. https://www.atlassian.com/agile/project-management/user-stories
7. https://www.atlassian.com/agile/project-management/epics
8. https://rubygarage.org/blog/clear-acceptance-criteria-and-why-its-important
9. https://dzone.com/articles/requirements-epic-feature-user-story-task-size-and
10. https://www.easyagile.com/blog/user-story-points/
11. https://dzone.com/articles/how-many-stories-per-sprint-rules-of-thumb
12. https://ancaonuta.medium.com/how-to-define-features-in-agile-methodology-2bd5039c67ff
13. https://www.c-sharpcorner.com/UploadFile/d9c992/the-agile-scrum-framework/
14. https://www.scrum.org/resources/what-is-scrum