

Basics of SCRUM

A modern way to run projects in an organization

Editor:

Romuald Lenarski

Authors:

Marta Matylda Kania, Izabela Pięta,
Julia Gutkind, Piotr Marchewka,
Katarzyna Wojciechowska

SCRUM glossary

Overview of **common errors**

Practical use of **SCRUM in organisations**



Wrocław 2023

Copyright © All rights reserved

Title

Basics of Scrum.

A modern way to run projects in an organization.

Authors

Marta Matylda Kania, Izabela Pięta, Julia Gutkind,

Piotr Marchewka, Katarzyna Wojciechowska, Tomasz Środa

Editor

Romuald Lenarski

Translation

Katarzyna Wojciechowska, Tomasz Środa

Photographs

Adobe Stock, System Firmbee

IFIRMA SA

Grabiszyńska 241G,

53-234 Wrocław

NIP: 898-16-47-572

REGON: 931082394

E-mail: contact@firmbee.com

www.firmbee.com

www.ifirma.pl

08

What is Scrum?

- 08 Scrum philosophy, theory and structure
- 09 Scrum Values
- 10 How to implement Scrum in your company?

12

Scrum Team: roles and responsibilities

- 13 Scrum Team composition
- 14 Product Owner
- 19 Scrum Master
- 27 Cooperation between Product Owner and Scrum Master
- 28 Development Team
- 32 Scaling Scrum

34

Scrum Artifacts

- 35 Sprint Backlog
- 37 Product Backlog
- 40 User Stories
- 47 Story Points and Estimation
- 52 Increment

54

Scrum Events

- 55 Sprint in Scrum
- 58 Product Goal, Sprint Goal and Definition of Done
- 60 Burndown Chart
- 66 Scrumban and Kanban boards in Scrum
- 69 Velocity in Scrum
- 70 Daily Scrum
- 72 Sprint Planning
- 73 Sprint Review
- 74 Sprint Retrospective

77% of high-performance teams use project management software.

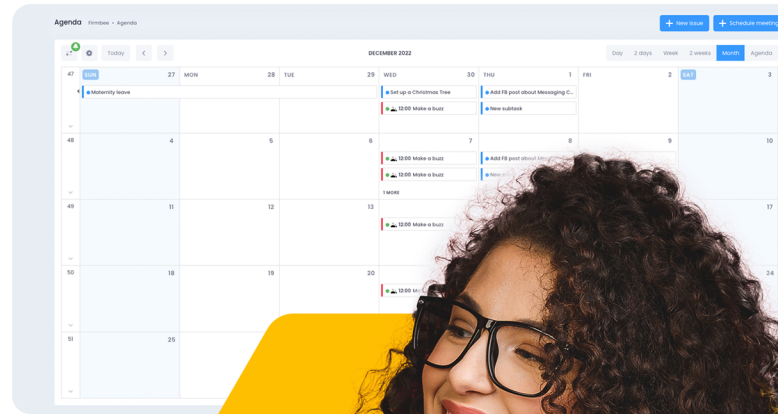
Join them with Firmbee!

- Plan and track the progress of your projects with **Kanban boards**
- Collaborate** with your team members
- Manage your **time and budget**

Discover more

A screenshot of a task creation form in Firmbee. The form includes fields for 'Event', 'Meeting', and 'Task'. The 'Task' field is selected and contains the text 'FB Stories for July'. Below this, there are fields for 'Add description', 'Characters left: 1000', and 'Place'. There is also a section for 'Assigned user' with a 'Contact' dropdown menu. Other sections include 'Attachments', 'Subtasks', and 'Comments'. At the bottom, there are buttons for 'Cancel', 'Log time', 'Share', and 'Save'.

A screenshot of a Kanban board for an 'HR Project' in Firmbee. The board is divided into four columns: 'New', 'DOING', 'READY', and 'IN REVIEW'. Each column contains several task cards with details such as task names, IDs, and dates. For example, in the 'DOING' column, there is a task 'Add FB post about payments' with ID 'A5782295' and a due date of '09.07.2022'. The board also shows a 'Project Risk' of 'Low risk' and various financial metrics like 'Budget' and 'Amount spent'.



The presented content is protected by copyright. Copying, processing and using it (in whole or in part) without the consent of its author/s is prohibited and constitutes a violation of the Law on Copyright and Related Rights

Scrum is one of the methods of Agile software development. This ebook will help you familiarize yourself with the basic knowledge of Scrum and then put it into practice. In the text, you'll find over 40 units, starting with basic Scrum terms and ending with useful tips and sources on where to gain more knowledge and experience in Scrum. You'll find out what are the roles in Scrum and who is responsible for each process, what are the Scrum Values, Scrum Artifacts, Scrum Events, and many more. Let's start your Scrum journey!

Scrum Glossary

Clear and plain language characterizes Scrum terms and jargon and function as the basis for communication and swift team performance during the process of project development. That's why we gathered the most essential explanations of the key notions of Scrum in the following glossary to introduce you to the topic that's gained significant importance in modern business management methodology.

Scrum basics

- **Scrum** – a kind of teamwork management system aimed at developing products and working under a defined scheme.
- **Scrum Values** – the 5 values underlying Scrum are: commitment to work, focus on goal, openness for change, respect and courage to solve problems.
- **Empiricism** – keeping planning and assumptions to a minimum in favor of relying on experience, observation, and experimentation.
- **Self-management** – a feature of Scrum Team which means that its members make their own decisions concerning what tasks to perform, when and how.

Roles in Scrum

- **Scrum Team** – a self-managing team consisting of the Scrum Master, Product Owner, and Developers, also known as the Development Team.
- **Product Owner** – a customer representative inside the Scrum Team. The one responsible for maximizing the value of the product by taking care of the Goal and its implementation by the Development Team.
- **Scrum Master** – a coach and leader of Development Team responsible for proper understanding of Scrum and acting according to its principles.
- **Developer** – each member of Development Team, regardless of professional specialization. A person responsible for co-creating useful Increment in each Sprint.
- **Development Team** – an interdisciplinary group consisting of all Developers involved in creating a Product.

- ◆ **Stakeholder** – a person who is not part of Scrum Team and has an interest in creating the best possible Product. Within the Scrum Team, he/she is represented by Product Owner and participates in Sprint Review.

Scrum Artifacts and their components

- ◆ **Artifacts** – Product Backlog, Sprint Backlog, Increment and their components. They are a reflection of the current state of progress on developing Product in relation to Product Goal, Sprint Goal, and Definition of Completion.
- ◆ **Product Backlog** – a structured list of work needed to create a specific Product, i.e., to achieve the Product Goal. Product Owner manages it.
- ◆ **Sprint Backlog** – a structured list of work required to deliver the Product functionality defined by the Sprint Goal. It is managed by Development Team.
- ◆ **Increment** – complete and valuable work performed by Developers in one Sprint. The sum of all Increments creates Product.
- ◆ **User Story** – a description of a partial functionality of a product from the customer's point of view. It takes the form of "As [user type], I want [what to do?] because [why? why?]".
- ◆ **Definition of Done** – placed in Product Backlog, a clear and transparent description of the expected state of Product after the completion of the Increment. It describes the work that has been done in the Increment.

Scrum Events and their components

- ◆ **Events** – a meeting that relates to the Scrum Team's work, its planning or feedback. These include Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective together with their components.
- ◆ **Sprint** – a recurring event involving work on a new version of a product. It takes place once or twice a month. It serves as a "container" for other Scrum Events and activities.
- ◆ **Sprint Planning** – a Scrum Team meeting where all the work to be done in the next Sprint is selected from the Product Backlog. It lasts a maximum of 8 hours.
- ◆ **Daily Scrum** – a daily meeting of Development Team where tasks for a given day are planned. It lasts maximum of 15 minutes and always takes place in the same place and time.
- ◆ **Sprint Review** – an event summarizing the completed Sprint in terms of Product Goal. For Scrum Team and Stakeholders. Its purpose is to evaluate the Increment and update the Product Backlog. Lasts 4 hours or less.

- ◆ **Sprint Retrospective** – an event summarizing the completed Sprint in terms of how the Scrum Team works. Its purpose is to improve Scrum Team performance. Lasts 3 hours or less.
- ◆ **Product Goal** – a description of the future Product that Scrum Team is working on. Product Backlog contains the written blueprint with instruction how to attain Product Goal.
- ◆ **Sprint Goal** – the work to be done within one Sprint expressed as a business goal. It provides consistency in the work of Development Team.
- ◆ **Burndown Chart** – shows the amount of work scheduled in Sprint or Product Backlog in relation to the time remaining to complete it.
- ◆ **Team Velocity** – an indicator to determine how much of the Product Backlog has become Incremental during one Sprint.

1. What is Scrum?

Scrum is the most popular way of modern, flexible teamwork management. Not only do software development companies follow its methodology but more and more often teams from various industries such as finance, marketing, HR and creative industries find Scrum practical and applicable. Its ever-growing popularity makes Scrum the most proven organizational framework adapted by teams striving for maximum effectiveness.

Scrum is simple by design. First, it facilitates breaking down difficult problems into smaller, more understandable components. Secondly, it provides solid structure by dividing the work into stages to execute in given time frames, bringing measurable and satisfying results. Thirdly, it enables planning of the next stages of work using obtained results and conclusions drawn from the ongoing processes.

Scrum philosophy, theory and structure



It is important to remember that **Scrum is only a framework. It provides guidelines to construct a detailed blueprint of actions to take together with adapted to the needs and capabilities of the team and the organization.** Despite its generality, Scrum is finely outlined. The popularity shows its effectiveness, as according to the 15th Annual State Of Agile report from 2021, Scrum principles are used by as many as 66% of teams working following the most modern methodologies. And this percentage increases to more than 80% of teams across disciplines if we add methodologies derived directly from Scrum.

Scrum is comprehensive and serves to optimize teamwork. It offers a clearly-defined starting point. Also, the generality of Scrum principles makes it impossible to simply apply them in an instant. However, the sketchiness of this framework is intentional and flows along with project management practice. The Scrum philosophy focuses on the need for continuous development and reshaping through feedback, reflection and experience. It rejects complex, rigid systems that organize work without taking into account specific realities. The authors of Scrum, Ken Schwaber and Jeff Sutherland, call this principle empiricism in the official Scrum Guide.

Scrum theory

Scrum theory's main principle concerns empiricism. **Empiricism means keeping planning and assumptions to a minimum in favor of relying on experience, observation, and experimentation.** It becomes possible and effective thanks to the iterative approach, i.e., working in short cycles, which includes not only working on the product, but also planning it and evaluating the results. Three pillars of empiricism are the most important for Scrum's effectiveness:

- ◆ **transparency** – thanks to it, both the people working and Stakeholders (to whom we will devote a separate entry in following article) can easily check the status of the work on Product at a given moment
- ◆ **inspection** – means frequent and reliable updating and checking the progress, thanks to which it is possible to detect problems and solve them quickly
- ◆ **adaptation** – means adjusting the ways of working and the Goals, which are described in a separate article, if there are errors or discrepancies during the inspection.

Empiricism works best if the team operating according to its principles has the ability to self-manage according to the lean concept. It implies a flexible organizational structure that allows adaptation to exist conditions, continuous improvement, and independence of Scrum Team.

Scrum structure

Scrum sets the framework for team action by defining: composition and roles in the team (Scrum Team), pace of its activities and meetings (Scrum Events) and methods to plan, manage and execute actions (Scrum Artifacts).

Scrum Team is an independent, interdisciplinary team of professionals working in Scrum, free from the influx of additional tasks from the organization. It is the basis of effective work in Scrum. Scrum Team consists of Product Owner, Scrum Master and Development Team. It is a small team with possibly constant composition, working on a specific Goal. Scrum Team should constantly improve and enhance not only the product but also its own way of working. This helps to increase the efficiency and quality of teamwork.

Scrum Team activities and meetings are called Scrum Events. These include Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective, and their components. The way of planning and executing, and the conditions for successful goal achievement, are described in Scrum Artifacts, i.e., Product Backlog and Sprint Backlog. These are very frequently updated documents reflecting the current state of work on Product.

Scrum Values

Scrum Values were not always present in the official Scrum Guide. They were added by user request to the edition published in 2016. We will describe the Scrum Values one by one, namely: **commitment, focus, openness, respect, courage**. The update of the Scrum Guide followed the principle of empiricism. That is, it was done based on the experiences of the teams implementing Scrum. It turned out that even the strict observance of the Scrum workflow didn't always cause positive changes in the teams. The purpose of introducing the Scrum Values was to clearly define the right approach to foster the implementation of Scrum principles.

1. Commitment

The first value that underpins the proper implementation of Scrum is the commitment to the work. A Scrum-oriented Development Team has to be committed to Sprint Goal and Product Goal, but even more important is the commitment of each individual Developer. It provides not only an active effort to solve problems that arise in the course of daily duties. It also includes the willingness to support other members of the Development Team when they need it.

2. Focus

If each of the Developers is focused on sub-tasks with full commitment, it can be easy to lose sight of the main Product Goal and even the Sprint Goal. As a result, the Increment produced in a given Sprint will turn out to be incomplete. Therefore, focusing on Sprint Goal is extremely essential from a team perspective.

Focus improves not only the level of tasks performed by individual Developers but also, their results come together to achieve Sprint Goal. And thus, the business value is realized: Scrum Team's workflow enables the creation and development of Product functionality.

3. Openness

The agile approach differs from the traditional approach primarily in its openness to change. That is why it was among the main values of Scrum. Both the Scrum Team and the Stakeholders benefit by cultivating the value of openness in their work. The variability in question can refer to many areas of the project, for example: reduction of the project scope, redefinition of Product functionality, reorganization of tasks performed by Scrum Team to optimize efficiency. Changes can result from Stakeholder decisions, business or technical constraints. Openness in the face of change is supported by transparency, which is one of the pillars of empiricism in Scrum. Thanks to transparency, Scrum Team members and Stakeholders know what the change results from and where it can lead.

4. Respect

Scrum Team members are independent, equal individuals. Developers, Scrum Master and Product Owner are all professionals who treat each other with due respect. There should be no room for hierarchy in a Scrum Team. Respect also serves to provide psychological well-being. This is especially true for members of the Development Team. By showing respect to each other, they can experiment freely and not be afraid to make mistakes, which are a necessary part of the innovation process and lead to improved Scrum Team performance.

5. Courage

Courage in problem-solving is, along with respect, a prerequisite for successful experimentation. It also makes the Scrum Team, when faced with difficult tasks, repeat attempts to solve them, until obtaining a satisfactory result. Being courageous in proposing business, technical, and interpersonal solutions helps to realize the innovation potential inherent in Scrum. Courage is also the foundation of self-management of the Scrum Team, which takes full responsibility for the consequences of the decisions made by the team members.

How to implement Scrum in your company?

Although the framework of Scrum is simple, its implementation in a company is not an easy task. Scrum is used to optimize teamwork, but initially, it can cause a lot of trouble. And even aggravate existing problems in the organization. So how to implement Scrum in your company?



Many entrepreneurs come up with the idea of using Scrum principles in their company. Scrum promises great team effectiveness, an energetic atmosphere, and no hierarchy. However, it sets very specific requirements for both the team and the individual people in Scrum Team. **When deciding to implement Scrum, it is worth considering the technical and organizational details.** So let's look at potential implementation issues, required team member competencies, and ways to separate Scrum Team from the overall organization.

1. Implementation. The task for Scrum Master

During Scrum implementation, the key figure is the experienced Scrum Master. He is solely responsible for the proper understanding of Scrum by all involved. Also, he assesses and amends all operations according to Scrum principles and values. Hiring Scrum Master with experience is critical if the company's employees have not worked with Scrum principles before. They will probably have a lot of questions, and Scrum Events will require detailed guidance by Scrum Master for at least the first few weeks. The Scrum Master's tasks are limited to the role of coach and leader, once the members of the future Development Team know the Scrum principles well enough. So, before starting other activities, answer the question: Are you able to find and hire the right Scrum Master?

2. Interdisciplinarity. How do you build a Development Team?

The next question a Scrum implementer must answer is:

Is my company already staffed with people capable of creating a self-sufficient, interdisciplinary team?

There are two problems that may arise during the creation of the Team: lack of sufficiently mature (senior) staff with complementary competences and rigid organization structure.

All members of Scrum Team should be specialists in their field. And their competencies should be complementary to each other. A well-composed, interdisciplinary team should not be dependent on the help of external specialists. This is especially substantial if the Team is working with confidential and sensitive data that should not be shared with people outside the organization. Using external help also disrupts one of the pillars of Scrum, transparency. It can also create the risk of creating hierarchies within the Team. For example, separating "second-class Developers", people who will not take full part in Scrum Team activities. **Problems with Scrum implementation may arise when the company is divided into strictly separated departments.** If each member of Development Team works in a different department – a great deal of reorganization will be necessary. One of the topics to think about is a common workspace for an interdisciplinary team.

3. Scrum Rhythm. Separating the Scrum Team

Another key issue to consider while implementing Scrum is to create a kind of "firewall" to protect the fresh Scrum Team from the influx of external tasks. It will probably be formed by people who worked on other projects in your company. By force of habit, people with whom the new Scrum Team members have worked will continue to seek their help. And this can generate conflicts, cause an influx of additional tasks, and disrupt the rhythm of Scrum Events.

Is it worth implementing Scrum?

If you are seriously considering implementing Scrum in your company and are aware of the problems that may arise, analyze again whether Scrum is definitely the solution for you. **"Yes" has been said to Scrum by as many as 66% of agile teams.** However, the statistics don't show how effective the teams are freshly after implementation. Nor how long it takes to get to and then surpass pre-Scrum efficiencies. What's more, paying a lot of attention to the size of the Scrum Team and the volume of projects the team will undertake.

2. Scrum Team: roles and responsibilities

Scrum Team is a team working according to Scrum principles. Its most important feature is the lack of internal hierarchy. Once the goal and scope of tasks to be done in a Sprint are agreed upon, team members take the initiative. From that moment on they make their own decisions about who, how, and when will do what. How does such a self-managing team work?



The Scrum Team as a whole is accountable for its actions. Therefore, it must have the authority to manage its work. This is especially significant in larger organizations. If the team is to operate following Scrum principles, the rhythm and manner of its activities cannot be disturbed from the outside. But of course, within a Scrum team, each person has their specific responsibilities.

The main task of the whole team is to work towards achieving Product Goal. The realization of this Goal is spread into short Sprints. In each Sprint, the Scrum Team builds a new working part of the project, the Increment. Small tasks performed by the team are well defined within a Sprint and distributed among the team members. A project and team management tool is useful in this task.

★ Issue details

event meeting **task** Generate To do In progress ⓘ

CRM integration

To understand CRM integration, we must first address the role of these systems in an enterprise. CRM systems are valuable assets for businesses across various industries. These systems hold customer data that can enable teams from marketing and customer service to sales and operations to better understand their customers how those customers interact with the business.

Characters left: 9629

Place

Assigned user Contact

Attachments +

Subtasks +

Cancel Log time Save

Source: Firmbee - Creating a new issue

Scrum Team composition

Typically, Scrum Team consists of no more than ten people. It should include open-minded professionals whose competencies complement each other. The composition of the Development Team changes depending on the ongoing project. However, there are always two key figures: Product Owner and Scrum Master.

1. Product Owner

The Product Owner is a team member equal in the hierarchy. But above all, he is the voice of the customer in the Scrum Team. **The Product Owner's primary responsibility is to communicate with the customer and then set priorities for the team's work.** This is reflected in both the Product Goal and the content of the Product Backlog. It is the Product Owner who answers questions about the Product asked by other team members by taking the customer's perspective. He therefore ensures that the direction and purpose of the Development Team is clear. He also accepts the work done by the Developers and approves the delivery of the result to the customer.

2. Scrum Master

The Scrum Master helps all members of the Developers team to understand the theory and practice of Scrum, and acts as an intermediary. Often they are at the same time members of the Development Team and the Product Owner's support. Still, only Scrum Master is responsible for the functioning of the team and its operations according to Scrum principles. They must watch over the team, to become both a leader and a coach. Scrum Masters focus on observing shortcomings and implementing upgrades and fixes. Additionally, Scrum Masters monitor the efficiency and effectiveness of Developers' work.

3. Developers

Developers are all team members who are not Product Owners or Scrum Masters. The name shouldn't confuse you – it's not just about programmers. Developers can be people with very different skills and responsibilities. Developers are responsible for planning their tasks, i.e., creating the Sprint plan contained in the Sprint Backlog and also the daily work of developing the Product in order to produce in each Sprint a useful Increment (increase) according to the Definition of Done.

Scrum Team operation

A Scrum Team is interdisciplinary in nature. This does not mean, of course, that everyone can perform every task set before the Scrum Team. Its members together have all the skills necessary to perform the task and manage their own work. That's why it's important to build trust between the people in the Scrum Team and also building trust by the organization to the Scrum Team as a whole and giving it independence. This is what will motivate its members to undertake more difficult tasks, beyond their daily responsibilities. Interdisciplinarity not only helps in making the team more productive. It also makes the professionals from different disciplines working as a Scrum Team able to help each other in completing the tasks set for the team, complementing their skills. This requires the skill of working together. Moreover, each team member has to possess general knowledge of what each person does. This way, Scrum Team participants know who to turn to for expert help if they have a problem with a task.

Product Owner



It may happen even though consisting of experienced specialists, the Development Team can't find their work swift and efficiently enough. When looking back at the big picture after the project terminates, it often turns out that it's the lack of a defined goal was to blame for that situation. When working under the Scrum framework, in order to prevent those and other troubling issues, the creators of the position of the Product Owner. **Only the Product Owner can make entries in the Product Backlog as well as make a final call in case of doubts concerning customer expectations.**

The basic responsibilities of this role include:

- **Collaborating with the Customer** – having regular conversations with the customer that lead to defining and specifying the features of the Product created by the Scrum Team; the primary goal here is to create a product that best meets the customer's requirements.
- **Articulating the Product Goal** – that is developing and defining the long-term direction of Scrum Team activities, and making sure that all team members understand it.
- **Keeping the Product Backlog** – Product Backlog is one of the Scrum Artifacts, defined in the official Scrum Guide:

"It's as an evolving, structured list of what is needed to improve the product. It is the sole blueprint of work undertaken by the Scrum Team. So let's take a closer look at the role of the Product Owner in the Scrum Team."

1. Voice of the Customer

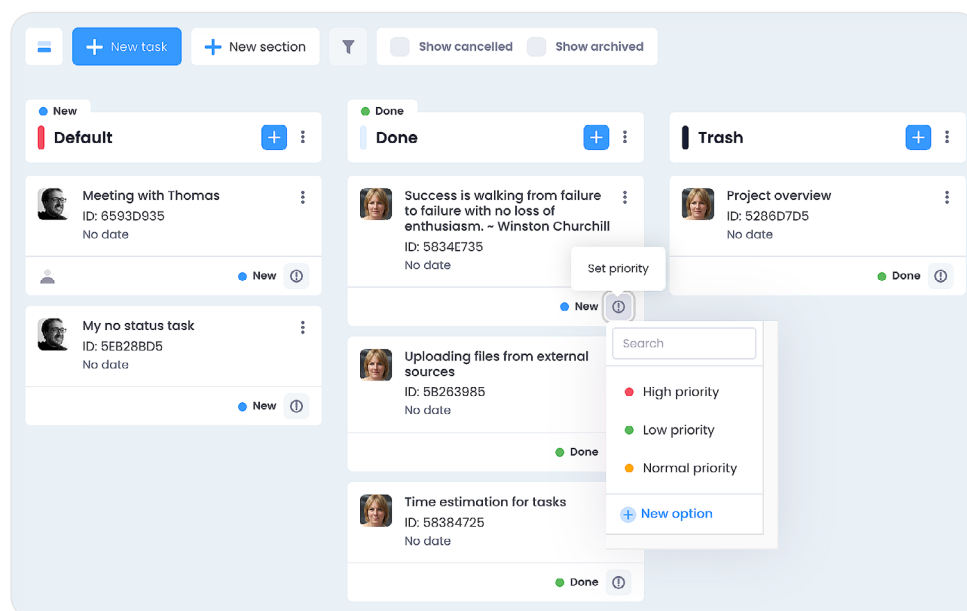
The Product Owner takes the angle of the customer in the Scrum Team. **Thanks to the presence of this role, the Scrum Team can tackle their doubts on an ongoing basis.** They don't have to interpret on their own the often unclear guidelines left by the customer. Taking decisions regarding the Product, establishing the priorities of the Development Team together with creating and organizing the entries in the Product Backlog the make the Product Owner is such a key member of the team.

The daily regular involvement of the Product Owner with the Development Team and the Scrum Master is crucial because of the short duration of Sprints. These are periods of intense work on the creation of Product Increment. There is no time to send questions to someone working in a different location and distance wait for an answer. Hence, it's the Product Owner who becomes a customer representative within the team. The team is present in it.

The Product Owners also attend all Scrum Team meetings to stay up-to-date with the progress of the Development Team. By recognizing the difficulties of co-workers, they can provide reliable information to the Customer and efficiently organize the work of the team. This occupation is to bridge the gap between the Scrum Team environment, i.e., the business environment, and the Developers. Also, to organize the work of the Developers Team as well as decide what are the criteria for completing the work on a given solution. And also approves of the moment of their fulfillment.

2. Product Owner and its purpose

The Product Owner is responsible for clearly defining and clarifying the Product Goal, i.e., defining the purpose of the team's work. In other words, this includes conceptual and organizational work, the purpose of which is mainly handling User Stories: creating the concept of the product and its functionality that correspond to and meet the Customer's needs. An equally large share of the Product Owner's duties concern management work – organizing and prioritizing the tasks in the Product Backlog.



Source: Firmbee - Set priority

From the Scrum Team perspective, it means taking care of the clarity of the activity horizon. We'll talk about the Product Goal in a separate section. Today we'll use an example:

Let the goal of the Scrum Team be to create a mobile app for organizing the team's work.

The Product Owner's task is to explain to the Scrum Team members what are the key functionalities of the application from the customer's point of view. For example – adding new team members by contact list or switching between the team and personal calendar views. Once the team understands the Product Goal, it will be up to the Product Owner to:

- 🟡 **keeping the right course** – so that always the most prominent functionality of the created application is in line with the Product Goal, i.e., organizing the team's work,
- 🟡 **clarifying more detailed issues** – searching together with the team, agreeing and specifying how the Product works,
- 🟡 **keeping an eye on priorities** – following the 'first things first' principle, the Product Owner's task will be, for example, to prevent the development of side or additional application functionalities which are of secondary importance from the customer's point of view.

Aquiring Customer or even Stakeholder vantage point makes Product Owners key decision-makers in the process of Product creation. What's more, they take part in discussions with people from outside the Scrum Team and carry the burden of valuable Product delivery to Customer.

3. Backlog guardian

The daily duties of the Product Owner include organizing the elements of the Product Backlog. This is a daunting task, as it keeps changing throughout the whole endeavor of the project. That written log contains all known measures and steps leading to the realization of the Product Goal. Of course, they are finely segregated and prioritized prior to their execution.

The Product Owner refines the tasks included in the Product Backlog and divides them into smaller. Then he decides which of them are ready to enter the implementation stage. On one hand, working with the Product Backlog is administrative and organizational work, which requires understanding the nature, capabilities, and limitations of the Developer Team's work. On the other hand, the Product Owner's task is to understand the nature, capabilities, and limitations of the work of the Development Team, as well as the external business conditions of Product development.

Yet on the other hand, the Product Owner's task is to create the Product Backlog according to the Scrum Team's needs. It must be clear, accessible, and understandable for all team members. On the other hand, the Product Owner is responsible for answering all the questions and doubts that Scrum Master and Developers have about the product. Thanks to him, the whole team knows what has already been done, what is in progress, and what still needs to be done.

Common mistakes of Product Owner

1. Problems between Product Owner and Customer

The Product Owner is the person who is personally responsible for the failures of the Scrum Team. Because of this position beyond the activities of the team, it is considered that the Product Owner is the single wringable neck. In other words, it is the Product Owner who suffers the most when the Scrum Team goes wrong. So how to deal with troublesome situations when they appear or better yet prevent them from happening in the first place? To answer that point, we've provided a clear and in-depth analysis of some major mistakes of Product Owners and Customers in the following table together with a detailed discussion of each.

Error	Problem generated	Suggestions for a solution
Inability to prioritize	Non-optimized Product Backlog, blurring of Product Goal	Listening, questioning, negotiating the Product Goal with the customer, carefully processing the results of the negotiation
Lack of assertiveness	Too many tasks for the Scrum Team to complete	Thinking realistically, knowing and remembering the capabilities of the team
Insufficient business skills	Risk of lowering the business value of the Product created by Scrum Team	Team Continuous learning and acquisition of business competencies

◆ Inability to prioritize

The mistake of not knowing how to prioritize is the bane of many Product Owners. Why is task prioritization a core competency? Because when everything becomes equally important, the Product Goal disappears. That is the intended effect of Scrum Team activity.

The problem starts already during the first conversations with customers about the Product Goal. The customer usually wants all his ideas to be realized as quickly and cheaply as possible. The Product Owner's task is to establish a list of priorities. His task is to create a list of clear and feasible expectations ranked from the most to the least important, based on unstructured customer expectations.

The problem with prioritization most often originates from misunderstanding the customer's expectations. It appears when the Product Owner is not able to extract information about the real Product Goals from the Customer. That is the answer to the question of what needs the product is supposed to respond to. So how do you protect yourself from this mistake? First – listen carefully to the customer. Second, learn to ask questions about the Goal and how each product feature works. Third – negotiate and limit the Goals to achieve. And for this, you will need assertiveness.

When the Product Owner has a list of tasks to do, proven methods can help prioritize and manage them. It can be helpful to have personalized to-do lists. In Firmbee, you can create to-do lists, set priorities, color-code statuses and track deadlines.

◆ Product Owner's lack of assertiveness

The problem that is closely related to the inability to prioritize is the lack of assertiveness. It results in inappropriately queued tasks and leads to blocking the realization of the Product Goal by compounding it with excessive tasks. Therefore, **the ability to say no to the customer is crucial.** Product Owner assertiveness should be based on three pillars: knowledge of the team's capabilities, knowledge of the solutions used and developed by the team and awareness of their role and value based on their place in the Scrum Team.

Therefore, one of the most important ways to prevent assertiveness problems is for the Product Owner to work with the Scrum Team daily. This will help him build realistic beliefs about the time and ability to implement the Customer's ideas.

◆ Insufficient business skills

The next mistake we would like to discuss is the lack of proper business qualifications. **The strengths of these Product Owners are usually specialized qualifications. Their competencies are more closely related to the area of the Development Team than to the business.** So there is a lack of well-established, practical knowledge about the competition, about the rules of the market, and the final customer of the product created by the Scrum Team. There is no simple remedy for it, as it may occur in very specific situations. Surely, however, a good course of action for a Product Owner is to acknowledge it and keep learning and gaining experience and business competencies.

2. Problems between the Product Owner and the rest of the Scrum Team

The ability to prioritize tasks, the assertiveness of the Product Owner, and his high business skills are the necessary prerequisites for creating an exemplary Product Backlog, the long-term foundation of the Scrum Team. If the Backlog is not outlined consistently and accurately, the problems in the Product Owner-Client relationship will spill over into the Product Owner-other Scrum Team member's relationship. And in turn, they directly affect the effectiveness of the Scrum Team. What other pitfalls await the Product Owner in his relationships with the other Scrum Team members? To make it easier, we have presented the problems between the Product Owner and Scrum Team in a table. Below you can find a detailed discussion of each problem and suggestions for solutions.

Error	Problem generated	Suggestions for a solution
Insufficient charisma	Development Team does not perform tasks included in Backlog, Product Owner's opinion is challenged	Building authority based on soft skills and knowledge
Insufficient specialized skills	Misunderstanding of the daily operations and capabilities of the Development Team	Orientation to the specialties of team members, as well as gaining knowledge of the Team's area of expertise
Dependence	Dilution of responsibility	Empowerment

◆ Insufficient charisma

On a daily basis, the Product Owner's job is to coordinate the Customer's guidelines with the way they are implemented by the Development Team. This undoubtedly requires having the right authority, listening skills and charisma. The problem of insufficient authority cannot be solved overnight. It requires long-term work on soft skills and also gaining knowledge about the scope of tasks and skills of other team members.

◆ Insufficient specialized skills

The role of a Product Owner is not strictly technical. However, knowing the basics of specialized skills of Development Team members can significantly increase the authority of a Product Owner. Insufficient qualifications in the team's area of expertise can not only generate problems with the charisma and authority of the Product Owner. The mistake of not being interested in what the members of the Development Team specialize in and the basics of their competencies can generate funny situations, but also situations with disastrous business and interpersonal consequences.

Therefore, **in order for the Scrum Team to deliver the best quality products, the Product Owner must have a thorough understanding of the product.** It should not be difficult to get the right qualification considering that the Product Owner is part of a team of professionals. They can provide not only explanations but also suggestions on where to get knowledge about their field.

Dependence

The Product Owner must be able to make decisions independently. Of course, the key issue is to know Scrum Team conditions and constantly communicate with the Development Team. However, it is the Product Owner who is held responsible for the effectiveness of his actions. For this reason, the Product Owners need to build their authority and take responsibility for the decisions they make. The final call on team direction, prioritization, and acceptance of tasks belong to them.

Scrum Master

Scrum Master is responsible for the daily operations of the Development Team. This role is on the borderline of tasks usually associated with the roles of a leader and a coach. That is why it is called a servant leader. At first glance, it might seem that Scrum Master is just an odd name for a project manager, i.e., the person who makes sure that the team members have all the skills necessary to achieve the goal.

He leads and debriefs Scrum meetings, and motivates and supports the team in their daily efforts to produce valuable Increment. However, there are a few key differences between a project manager and a Scrum Master.

1. Servant leader

First, a project manager in traditionally managed organizations is expected to withdraw from the team after checking that everything is in order. He will come back after the agreed time to check if the tasks are done. This is not the case with the Scrum Master. **The Scrum Master works with the team at all times.** Just like the Product Owners, Scrum Masters are irreplaceable and ever-present members of the Scrum Team and their participation in the daily work is essential. It is their job to maintain the morale of the team.

The second issue that strongly distinguishes a project manager from the Scrum master is the issue of solving the problems encountered during teamwork. In the traditional approach, the team should deal with them by itself. In Scrum, on the other hand, if there are problems, the Development Team should ask for help from the Scrum Master. Here, this role is most closely similar to the one of a coach.

The Scrum Master is also a coach when building the motivation of the whole Development Team and each individual. One of the Scrum Master's key tasks is to encourage individual team members to develop themselves. He can point out skills that are worth improving or suggest new, interesting development paths. First, he can recommend new captivating development paths, especially the ones that increase the interdisciplinary character of the team. Expanding the competencies is aimed at increasing the effectiveness and harmony of the Scrum Team, which results in greater internal controllability and better self-management capabilities.

The Scrum Master's task is also keeping the team focused on task execution and, closely related to that, protecting the team from the influx of external tasks. If the Scrum Team works as a part of a bigger organization, its work may be disturbed by the urgent needs of other units. It is the Scrum Master's job to explain to others that it is not good to disturb the Scrum Team's work rhythm. The check-ins function available in the project management system can be helpful in this task. Using it, the Scrum Master can send automated questions to team members and verify the answers received periodically.

Check-ins
✕

What question do you want to ask?

What have you worked on this week?

Characters left: 150

How often do you want to ask?

On selected days
 Once a week on
 Once a month on the first

On which day?

Mo
Tu
We
Th
Fr
Sa
Su

At what time of day?

Beginning of the day (9:00am)
 End of the day (4:30pm)
 Let me pick a time...

Who do you want to ask?

Nothing selected ▼

Add all project members

Cancel
Save

[Source: Firmbee - Check-ins](#)

2. Scrum Guardian

The key competence of a Scrum Master is an excellent knowledge of Scrum principles. And the key task – making sure that Scrum recommendations are followed. It helps all individuals to understand the theory and practice of Scrum. **An important part of the Scrum Master’s duties is to present the Scrum Team’s way of working to the Customer and other Stakeholders.** This is often the key to achieving the Product Goal. The Customer needs to be aware that in Scrum, as in other agile methodologies, continuous collaboration with the Customer is more indispensable than contract negotiations. This means, among other things, consulting the Product Owner about the project priorities instead of preparing the specification once.



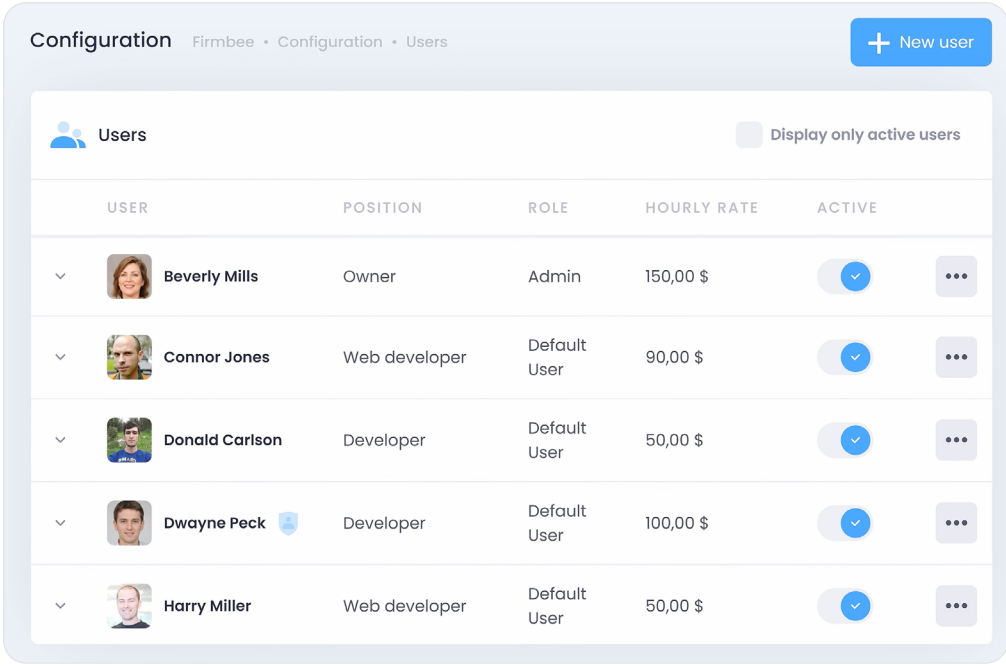
Since the Scrum Masters are the guardians of Scrum, they can also be its evangelist in the organization. However, their tasks primarily involve training the team members. **Scrum Masters make sure that the Scrum rules are clear and understandable to everyone in the Scrum Team.** They are also responsible for all the Scrum Events to take place. What’s more, it is also the Scrum Master’s job to moderate these meetings, monitor their productivity and ensure they stay within the designated timeframes. The key role is to support the team in reaching the Goal. In other words, to make all meetings constructive and productive.

Characteristics of a good Scrum Master

A good Scrum Master - meaning what kind? Definitely one who helps the team perform in the best possible way. He achieves this mainly by maximizing the internal potential of the team and removing obstacles that stand in the way of smooth implementation of the Increment. This requires the right outlook, approach to the problems encountered by the team and soft skills.

1. Planning and coordinating

The ideal Scrum Master is a person who could organize work brilliantly. This allows him to organize the Product Backlog efficiently and on an ongoing basis. However, the Scrum Master's organizational skills are primarily used to coordinate the work done by others. That's why it's good for him to be as efficient in organizing Scrum Events, as in moderating them. His tasks also include filling in missing information and answering Developer questions and facilitating access to training and tools for team members. Scrum Master is primarily concerned with working with people. By managing the team and ensuring smooth operation, each Scrum Team member knows their role scope of duties within the team. In other words, everyone knows what to do and why. With a database of contacts in the project management system, each user can be assigned a role, a job title or the rate of pay.



The screenshot shows a user management interface with the following data:

USER	POSITION	ROLE	HOURLY RATE	ACTIVE
Beverly Mills	Owner	Admin	150,00 \$	<input checked="" type="checkbox"/>
Connor Jones	Web developer	Default User	90,00 \$	<input checked="" type="checkbox"/>
Donald Carlson	Developer	Default User	50,00 \$	<input checked="" type="checkbox"/>
Dwayne Peck	Developer	Default User	100,00 \$	<input checked="" type="checkbox"/>
Harry Miller	Web developer	Default User	50,00 \$	<input checked="" type="checkbox"/>

[Source: Firmbee - Database organisation](#)

2. Leading by guiding

Scrum Master's role has certain defined characteristics. Such as acting as a guide, a person leading the way to safety but restraining from imposing superiority. Like a facilitator, Scrum Master helps by pointing out ways to increase productivity and improve collaboration. In other words, a good Scrum Master doesn't devote time to force decisions or solutions concerning the Product the team is creating. **Instead, the main focus revolves around the team performance, by doing so, find effective solutions and suggests how to work together.**

That is why it is so important for him to be able to ask questions. Questions that are not traditionally associated with the role of a manager. That is, instead of checking tasks and asking – were they done correctly and on time? And why not? The good Scrum Master asks:

- 🟡 How can I help you?
- 🟡 What could help the team make progress on this project?
- 🟡 Did we miss something?
- 🟡 Have we covered all the issues?
- 🟡 What else could go wrong?
- 🟡 How much time is needed for this?
- 🟡 Will we continue this approach in the next iteration?



Asking questions is not enough though. The key is the ability to listen and benefit from the knowledge, insights and assistance provided by the team.

3. Persuading with care

The basis of a Scrum Master's strong position in a Scrum Team is to treat all team members equally.

The key is to make sure that every member of the Development Team feels their contribution to the team is valued. The next step is to empower team members by supporting their professional and personal development. A Development Team composed of equal and confident individuals will strive to solve problems independently.

However, this means a much more difficult way of management than the traditional one. A Scrum Master should prefer dialogue and discussion over forcing, and guiding and asking questions over finding solutions for the Developers. This requires a lot of patience but also decisiveness. After all, it's not about leaving contentious issues unresolved. **That's why Scrum Master has to master the skill of influencing the team. This requires high emotional intelligence.** He usually doesn't work in a position that would make him, in the traditional sense, the superior to the Development Team.

4. Seeing the big picture

The Scrum Master works inside the team, and does not look at the team's work "from above". However, this does not apply to all situations. By not being directly involved in the creation of the product, the Scrum Master can see patterns and dependencies that are invisible to the members of the Development Team.

This different perspective is also good for motivating others. It allows to set horizons for development and consistently supports the members of the Development Team in reaching them. The ability to have a broader perspective of the team's work will make it possible for a Scrum Master to suggest a systemic change in the approach, instead of helping the Developers to fix single, recurring mistakes. Energetic, another quality of a good Scrum Master, who can energize the team, also helps to support the new vision.

Common mistakes of Scrum Master

The work of a good Scrum Master can be recognized by the fact that at some point they are no longer needed in the daily work of the Development Team. However, this is not always the case. What are the reasons for the mistakes of Scrum Master? A Scrum Master's job is primarily to support the work of the Development Team. Therefore, the most common mistakes of Scrum Master usually stem from the way he participates in the daily functioning of the Developers. We've divided these mistakes of Scrum Master into two groups. The first one includes problems resulting from too much involvement, while the second one includes problems resulting from the insufficient presence of the Scrum Master in the life of the Development Team.

1. Too much control

The need to maintain too much control over the Team often causes mistakes in the application of Scrum. Scrum Master errors most frequently become apparent in the following situations.

- **The Scrum Master looks for a solution to the problem instead of helping the team deal with the difficulty.** Typically the root of the problem is that the Scrum Master is also an expert in what the Development Team is doing. Their inability to step out of the expert role makes them unable to effectively assist the team in finding solutions on their own. This approach can also lead to one-person, authoritarian decision-making – and this is probably the biggest mistake a Scrum Master can make.
- **The Scrum Master does not allow the team to make mistakes. This problem is closely related to the previous one.** If the team is effectively protected by the Scrum Master from making mistakes, it will not learn to solve problems on its own or take responsibility for its work. It will always rely on the advice and expertise of the Scrum Master.
- **The Scrum Master tries to change people instead of working on the team atmosphere.** This problem includes too much emphasis on changing the behavior of a team member or members, as well as personnel changes. It is a mistake to change the composition of the Development Team while working on a Product Goal if it is not absolutely necessary. It can introduce significant delays in its realization, and disturb the rhythm of work of the Development Team and also disrupt the rhythm of the Team formation.
- **The Scrum Master acts as the supervisor of the Development Team in the organization.** This is a mistake that does not often result from the Scrum Master's own decisions. However, it can exacerbate all the errors that arise from the need to control the Team.
- **Scrum Master over-involves himself in the operation of the Team.** When the Team is composed of experts who know each other's skills and responsibilities and is functioning according to Scrum principles, the Scrum Masters should not interfere uninvited with the way the Team works. If they do, they are simply interfering with the smooth running of the team. Good Scrum Masters, thanks to their well-established position as a coach and leaders, will be asked for advice in emergency situations or situations requiring a fresh look. That's why they should be available on call for Developers without not imposing their presence.

- **A Scrum Master is too rigid in his adherence to Scrum principles.** If any aspect of Scrum is not working in a particular Team, the Scrum Master should try a different approach. Every Team is different, and Scrum is just a general framework.

2. Too little commitment

Not only too much but also not enough Scrum Master's involvement can lead to many mistakes. We have described the most common ones below.

- **The Scrum Master is insufficiently familiar with Scrum principles.** This mistake will most likely lead to their improper implementation. And the Team's work will only seemingly be Scrum work.
- **The Scrum Master is not enforcing the Scrum principles.** The Scrum Master's inadequate day-to-day presence means that he is not protecting the team as he should. This can lead to a lack of protection from the influx of outside tasks or to the failure of the Development Team to meet the Sprint Goal.
- **The Scrum Master does not make sure that a consistent Scrum rhythm is followed.** Carelessness in organizing Scrum Events can lead to wasting time. This will result in too long or poorly run events – Sprint Planning, Sprint Retrospective or Sprint Review (which we'll write about in separate posts). It is also a mistake to postpone events or change their duration.
- **Scrum Master does not respond to conflicts in the Team.** Expecting conflicts on the Team to resolve themselves over time is a Scrum Master's mistake. Conflict is not always bad, but the Scrum Master should not only be aware of its existence and current state but also engage in it as a negotiator and also be able to use the conflict to change and improve the Team.
- **Insufficient Scrum Master presence.** The problem arises when the Scrum Master spends too little time working with the Team and gets involved in specialized tasks, for example. This makes him listen too little and ask too few questions. This, as we wrote in the previous article, is a key skill for a Scrum Master. The result is that the Scrum Master doesn't know well enough what is the current situation and atmosphere in the Team and he is content with the status quo.
- **The Scrum Master does not question the status quo.** In order for the Development Team, and the Scrum Team as a whole, to grow, it is necessary to constantly challenge the status quo. This is often a risky and potentially infictive activity. A Scrum Master should undertake it with the awareness of the difficulties he may encounter. However, there is no such thing as a "mature Development Team that is no longer evolving". Leaving it alone will quickly lead to a significant deterioration in its performance.
- **The Scrum Master does not share his observations of the Team's performance with the Team.** Keeping this knowledge to themselves makes it difficult, or even impossible, for the Team to grow. While completely focused on daily responsibilities, Scrum Master does not work on the way the team members work together. This frequently leads to the accumulation of problems and conflicts.

Statistics and metrics the Scrum Master should track

Why does the Scrum Master need statistics and metrics? First, to check if his methods of working on the predictability of results and improving the effectiveness of the team are effective. But also to keep track of how their actions affect the Development Team. That is, how they shape the employee user experience (EX). So we present the statistics and metrics that the Scrum Master should track.

1. Measuring the results of the Development Team's work

The most commonly used statistics and metrics the Scrum Master should track are those that describe the pace and flow of task execution. These are the Burndown Chart, the Burnup Chart, and the Cumulative Flow Chart. These measures both product development and team effectiveness. Each allows you to approach these issues from a different angle, so it is a good idea to show them together. They are handy tools to assess progress on different scales, during a Sprint as well as the entire product development process.

◆ Burndown Chart

The Burndown Chart shows the Scrum Master and the Development Team how much work has been done and how much is left to do. The X-axis shows the time left to complete the work. The Y-axis shows the amount of work left to be done that was planned in the Sprint Backlog or Product Backlog. This chart also helps to determine the Speed of the Development Team. It is an average amount of work done during one Sprint. This simple tool enables the Scrum Master to not only see how efficiently the team is working. It also helps to answer questions:

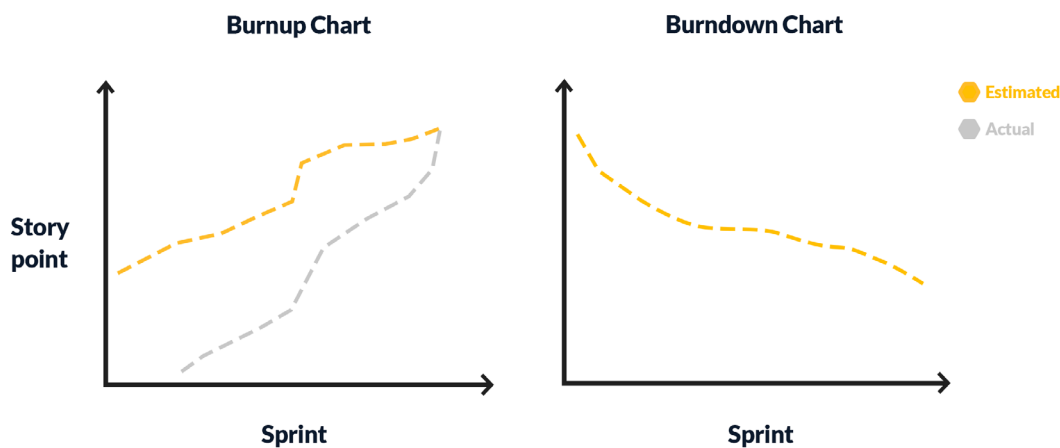
- ◆ What part of the work has already been completed?
- ◆ How many tasks are left to complete?
- ◆ How long will it take to develop the Product?



When using the Burndown Chart, the Scrum Master needs to keep in mind that it is not the only tool to statistically assess the team's progress. **It works best for projects where the scope of work is fixed and known.** It doesn't perform well with creating very innovative solutions with a new Client. Then the amount of work to be done in the whole project – that is the content of the Product Backlog – can change significantly during the project, making it difficult to use the Burndown Chart.

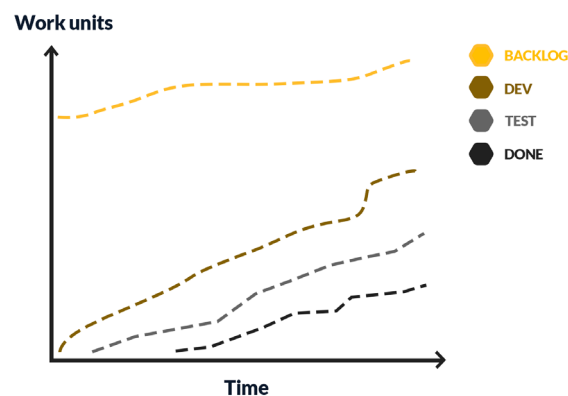
◆ Burnup chart

The Burnup Chart is the reverse of the Burndown Chart discussed above. Here too, the Y-axis shows the amount of work remaining to be done. The X-axis, on the other hand, shows the completion time expressed either in the number of Sprints or in dates. However, the Scrum Master uses Burnup Chart for a slightly different purpose. **This is because it not only aids you in measuring the progress of the product and the progress of the Team.** This metric also assesses how the scope of work in a project changes over time. Therefore, it works well for projects with variable scope. The Burnup Chart is also a planning tool that is becoming more effective over time. It provides answers to the question of how much work the Development Team is estimated to do in the next Sprint.



◆ Cumulative Flow Diagram

The third type of diagram that is very fruitful in the Scrum Master's work with the Development Team is the Cumulative Flow Diagram. It features the analysis of how stable is the pace and productivity of the Development Team. The layout of its axes is the same as the Burnup Chart, so it is often referred to as its more complex version. However, the Cumulative Flow Diagram is not only to determine the number of tasks completed in a given time period. It also takes into account the number of tasks which are waiting in the queue for execution. Thanks to this, **it enables to diagnose the so-called "bottlenecks" – moments of the process which slow down the creation of a product.** This very diagnostic feature makes it one of the most useful metrics in the hands of the Scrum Master. This is because it allows to reorganize the work in a way to distribute the strength of the Development Team differently and avoid downtime.



2. Monitoring employees user experience Developers

Regular and meticulous maintenance and analysis of statistics is an essential part of an effective Scrum Master's work. However, he has to keep in mind first the employee user experience of the Developers, i.e., the way they perceive the work in the Scrum Team. However, it is not the quality of the metrics that decides, but the way in which the Scrum Master uses them. If the statistics are kept in accordance with Scrum's principles – they are transparent, public, and understandable to the Developers involved – they can be a way to motivate the team to work more efficiently or to reward them for great results. **However, statistics can function as a tool to put pressure on the Development Team.** Then their indications become a generator of accusations and resentments. They can contribute to deteriorating team morale and spoiling teamwork practices.

The second important factor of employee experience of Developers, which the Scrum Master working with statistical tools has to take care of, is the way of managing their time. This is because the Scrum Master needs to have enough time to take care of the Development Team. For this reason, in case of a large project, it is worth considering including an additional person in the Scrum Team. He will act as a project manager and will take care of the metrics. Thanks to this, it will relieve the Scrum Master – and to some extent the Product Owner – from the tasks that distract him from working with the Development Team.

Cooperation between Product Owner and Scrum Master

The Product Owner puts a clearly defined Product Goal in front of the Development Team and requires progress in its implementation. The Scrum Master takes care of the quality of the process of its creation: a good atmosphere accompanying the Team's work, motivation and removing obstacles. However, both the Product Owner and the Scrum Master are not two independent forces acting on the Development Team. Although the way each of them works is very different, their interests converge: it is all about the Scrum Team working effectively. That's why the relationship between Scrum Master and Product Owner, and their effective collaboration, is so important. Most of the Product Owner and Scrum Master's tasks revolve around their responsibilities related to the work of the Development Team. However, **the Scrum Master's duties and responsibilities also include supporting the Product Owner's work.**

1. Supporting effective communication with Developers

Effective communication between the Product Owner and the Development Team requires at least two strong fundamentals: comprehensibility and sufficient impact. The Scrum Master helps the Product Owner to strengthen them.

◆ Understanding the Product Backlog

One of the primary ways the Scrum Master helps the Product Owner is to make sure that the formulated messages are understood by the Development Team. The Scrum Master reviews the Product Backlog entries and asks additional questions to improve their clarity by paying attention primarily to:

- ◆ **clarity of entries** – so that Developers know exactly for what purpose they are developing a given functionality,
- ◆ **keeping entries concise** – so that descriptions of planned functionalities include only necessary information and it takes as little time as possible to read them.

In this way, the Scrum Master prevents a discrepancy from arising between the Product Goal, as the Product Owner imagines it, and how the Development Team members understood their task.

◆ The power of the Product Owner's influence

Scrum Master helps the Product Owner to enhance message effectiveness and charisma. The Scrum Master acts as a coach with whom the Product Owner can discuss problematic issues concerning the Product and its realization. That is why individual meetings during which discussions take place between them are so important. **Thanks to these discussions the Product Owner can clarify the product vision and answer the Scrum Master's questions before presenting it to the team.**

The Scrum Master, by giving feedback, makes the Product Owner's message during a meeting with the team stronger and clearer. This necessary preparation helps the Product Owner communicate the Product Goal better and better during Scrum Events.

2. Knowledge from experience

The Scrum Master also helps the Product Owner to realistically plan the tasks for the Development

Team. It may happen that a well-prepared Product Backlog doesn't match the organization's way of working in which the Product Goal is to be realized. The Scrum Master will therefore support the Product Owner with knowledge from experience by drawing it from the observation of failures and difficulties that occurred in previous projects. Thanks to the empirical knowledge, Scrum Master can foresee the difficulties in performing the tasks and achieving the Product Goal that results from the specifics of the organization, the Team, or its specialization.

3. Introducing Stakeholders in Scrum

The Scrum Master daily works mainly with the Development Team and sometimes also with the HR department, especially during the team-building process and in those rare moments when the team needs to be extended or changed. The Scrum Master's daily duties usually don't include the Stakeholders' cooperation – this is the Product Owner's job. The exception is when you start working with Stakeholders who are not familiar with Scrum principles and roles. This is when the Scrum Masters work with the Product Owner in meetings with all the people involved in the creation of the Product. They explain who is who in the Scrum Team and also help the Product Owner to implement good communication practices. These include, for example, the active presence of Stakeholders during Sprint Review or creating good User Stories.

Available in Firmbee, the Contacts tab stores and organizes stakeholders' data in one place. The Product Owner and Scrum Master can easily check the documents, payments and projects associated with a contact and tag regular and potential customers with the available labels.

Development Team

The Development Team working in accordance with Scrum principles is an independent group of specialists. It does not use the support of external specialists or subcontractors. But what determines that the Team is well matched to accomplish the Goal? And what responsibilities are included in the tasks of a Development Team – regardless of its specialization?

Development Team features

In order to be effective, a Development Team must have at least three characteristics: the ability to self-organize, the pursuit of development, and interdisciplinarity.

1. Self-organization

When we talk about Scrum Team, of which the Development Team is a part, we use the term "self-management". It means self-management at the organization level. The Scrum Team as a whole decides not

only who will do the work and how, but also what they will work on. In a Scrum Team, a large part of the management tasks belongs to the Product Owner and Scrum Master. Therefore, in the case of a Development Team, self-organization is more important than self-management. **It refers to planning responsibilities, i.e., deciding for yourself who will perform certain tasks, when and how.**

2. The pursuit of development

A key feature of an effective Team is the drive for growth. **The way of completing the tasks set before it should be ambitious.** This results not only from individual predispositions and attitude of each member of the Development Team. Raising competence and effort is also encouraged by the atmosphere in the Team, which defines it as a whole.

3. Interdisciplinarity

Interdisciplinarity of the Team means that its members together should have all the skills necessary to create valuable Increment in each Sprint. It also means that each member of the Team performs the tasks necessary for that Sprint. Everyone does what is necessary to achieve the Goal. Even if it means taking on new tasks beyond the Developer's expertise. It is a mistake to rigidly stick to one's professional competencies or role.

Development Team composition and responsibilities



According to the Scrum Guide, the maximum number of Developers is eight. Such a small composition encourages communication and openness, as Team members have the opportunity to get to know each other. However, the Team should not be smaller than three people. It needs to be big enough to make business-visible progress in each Sprint. **Developers within Scrum are called people with a wide variety of skills and responsibilities.** In no case is the name reserved for people who do programming. Thus, the Team may include programmers and designers, researchers and analysts, testers and scientists, as well as other specialists.

There is no hierarchy among Developers. That is why they do not use professional or scientific titles. An important assumption about the composition of the Development team is that it is a unity. Therefore, smaller teams working on other Goals should not be separated from it.

Development Team responsibilities

The responsibilities of the Development Team can be divided into three areas. These are: planning tasks, working on the product, and improving collaboration within the Team.

1. Planning tasks

Task scheduling is an obligation that all Scrum-based Development Teams have to fulfill. It consists in creating a Sprint plan and putting it into a Sprint Backlog. The most significant thing is that the Development Team works on it together. This way each of the Developers will be able to realistically determine the number of tasks to be done in a given Sprint. In the long run, this allows the Team to

maintain a constant pace and plan more accurately. It is equally essential to keep an eye on the pulse, i.e., to adjust the plan to reality daily. If problems arise, there may be a need to change: to reorganize the tasks, distribute the work differently, or talk to the Scrum Master about emerging difficulties.

2. Working on the product

The forms of working on a Product can vary dramatically depending on the area in which a given Development Team operates. Generally speaking, the goal to be achieved in each Sprint is to create an Increment, i.e., a business-valuable Product feature. It is useful here to speak directly and apply the following rule: *“When you undertake work on a Product, you must leave it in a state that is not only improved but no less finished than the previous version.”*

Applying this principle means that the Team as a whole takes responsibility for the Increment. If a Developer performs tasks carelessly, causing the quality of the Product to deteriorate, someone else will have to do the work for them. On the other hand, if any Developer hits bugs in the Product, they should fix them themselves or pass the bug information to someone who can do it.

3. Improving Collaboration in the Team

Working on the way the Team operates is about constantly improving the efficiency and effectiveness of individual Developers. However, it is also, or maybe above all, work on communication between Developers. **The improvement consists in working out solutions that enable efficient and accurate task division and also practicing skills:**

- 🟡 **criticize solutions, not people** – changing the language we use to describe work leads to a change of attitude and improved collaboration,
- 🟡 **distancing yourself from your ideas** – it enables humor and more honest feedback,
- 🟡 **building trust** – thanks to trust there can be many more innovative ideas proposed by Developers without fear of negative reaction of the environment.

Improving Team collaboration is accomplished through ongoing reflection on how the Team works and providing feedback during the Scrum Events.

Common mistakes of Developers

Many of the mistakes of Developers working in Scrum have their origin in their approach to teamwork. On the one hand, it is misunderstood independence and defending one's ideas against the team's interests. On the other, it is relying on others and the lack of independence. Another source of problems may be a misunderstanding of team responsibility.

1. Being overly attached to your ideas

Developers' daily responsibilities include finding innovative solutions to complex problems. The effort put into developing solutions can cause them to become overly attached to their ideas. This in turn makes them lose sight of the Product Goal and spend too much time developing side solutions that are not useful from a business perspective. And they are also less willing to look for alternative solutions, which threatens the agility of the Team.

2. Self-employment

If any Developer has difficulty understanding their role on the Team, they will try to separate their tasks from the Sprint Goal. Worse yet, they will do them without reference to the rest of the Team. It can also become a problem if they arbitrarily make changes to the Sprint Backlog. This is how the misunderstood independence of one of the Developers can stem from communication problems.

An excessive desire for independence can be rooted in a lack of recognition for a Developer's individual accomplishments. It appears when his or her contribution to the work done by the Team is evaluated out of proportion to the effort put in and the difficulty of the task. Working on your own can become a source of serious conflict within the Team. That is why it is so important for the Scrum Master to react and solve the underlying problem as soon as possible. This is because it may turn out that the mistake does not lie with the Developer, but with an incorrect assessment of their involvement.

3. Withdrawal of Developer

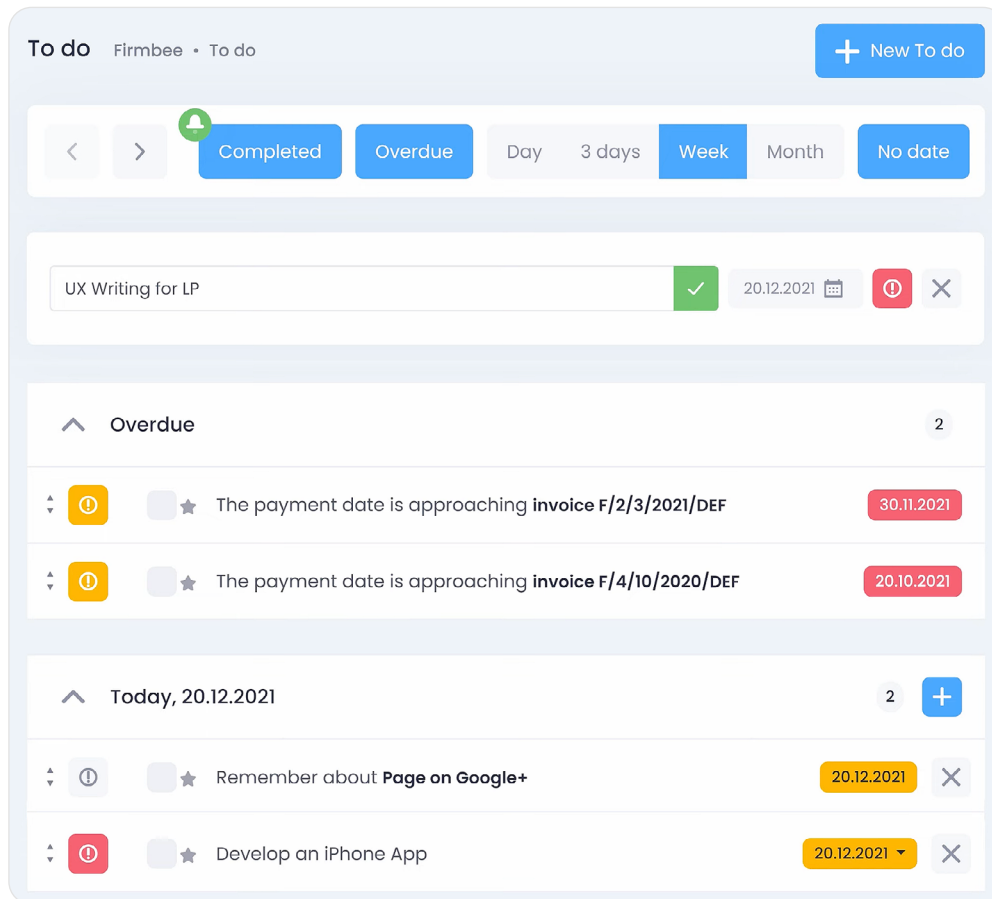
The problem resulting from the previous two – working on your own and being overly attached to your own ideas – can be a problem of lack of communication. Then those Developers start to isolate themselves from the Team. Although they perform their tasks according to the Sprint Backlog, they withdraw from the life of the Team. In such a situation the Scrum Master should pay special attention to the withdrawn Developers. Appreciate their contribution to the Team and encourage them to adopt a proactive attitude.

4. Independence

Self-organization is a characteristic of a mature, well-composed Development Team that we described in a previous article. It means that despite difficulties, Developers do not rely on other people to tell them how to distribute tasks among themselves, how and when to complete them. However, self-organization can give rise to interpersonal misunderstandings. In such a case, it is necessary to have the Scrum Master present at all times to make sure that the tasks that need to be done to achieve the Sprint Goal are distributed. This is when the problem of dependency of Developers arises. Again, the Scrum Master should come to the rescue by encouraging the Development Team members to be self-determined and take responsibility for their tasks.

5. Limiting responsibilities to the scope of authority

Another problem that Developers have to face, especially in the forming Team, is the unwillingness to perform tasks other than those belonging to the Developer's core competencies. This mistake can lead to a significant reduction in the effectiveness of the Development Team. Not all Sprints utilize the core competencies of each Team member. Therefore, they must be open to performing other, ancillary, or organizing tasks that are equally relevant to the Sprint Goal. In order to better organize their work, Team members can use a project management system, and more precisely, a to-do list, where they can write down their responsibilities, assign statuses, as well as set deadlines and priorities.



[Source: Firmbee - Creating a to-do list](#)

6. Sprint Backlog clutter

One such task is keeping the Sprint Backlog in order. It is a key task for the smooth operation of the Development Team. However, a common mistake is shifting the responsibility for keeping it between Developers. This hinders not only the work on the Sprint Goal but also the development of the Team and its ongoing improvement.

Scaling Scrum

As soon as an organization grows, new kinds of problems appear. For instance, a drop in employee effectiveness that's caused by complex internal structure, difficult decision-making or direction setting. Companies operating agile at the small project-team level often look to scale up.

Many enterprises do well without scaling Scrum. Even if many Scrum Teams are running simultaneously, they don't need coordination as the groups operate independently. However, this does not mean that it is a multi-team Scrum. **The need for scaling comes only when most of the organization is working on one product and can synchronize its multiple Scrum Teams effectively.** Most organizations that adopt agile management methods at scale choose the SAFE model, or Scaled Agile Framework. Today, however, we won't focus on SAFE but we'll discuss a different model called Scrum@Scale, as according to the 15th State of Agile report from 2021, it's the second-best choice among businesses that opt for agile.

Scrum@Scale

In 1996, the creators of Scrum, Jeff Sutherland and Ken Schwaber, were working on a large project. As they were doing it, they were having trouble keeping smaller teams working in Scrum in sync. They came up with a way to scale it, which they eventually called Scrum@Scale. Analogous to the official Scrum Guide was the Scrum@Scale Guide, which defines this way of scaling work as: “A framework within which networks of Scrum Teams operate following the Scrum Guide to solve complex adaptive problems and creatively deliver products with as much value as possible.”

The basic premise of Scrum@Scale is simplicity and efficiency. Therefore, its operation is based on a scale-free architecture. In other words, it uses Scrum to scale Scrum. In such a way, a Scrum Team composed of individuals acting as Product Owner, Scrum Master or Developer becomes the Scrum of Scrums: a team consisting of teams.

The Scrum of Scrums is a Scrum Team with people taking traditional Scrum roles. However, since the task of Scrum of Scrums is to integrate the results of the work of several Scrum Teams, it needs additional posts:

- **Product Owner Team** – a group of Product Owners who meet to agree on priorities and create a cohesive product vision
- **Chief Product Owner** – Scrum Team’s Product Owner or a person who deals exclusively with the Scrum of Scrums
- **Scrum of Scrums Master** – the person who oversees the effectiveness of the Scrum of Scrums. They meet at the same Scrum Events and use similar Artifacts.

Further scaling and Scrum@Scale issues

The scale-free architecture of Scrum@Scale means that it enables scaling more than just once. **If an organization needs to coordinate teams on an even larger scale, it can set up Scrum of Scrums.**

However, scaling Scrum, like any other management methodology has its flaws, and in this case, they are similar to those of the basic Scrum Teams, only they are proportionally greater. That’s why we recommend to work out the details of the collaboration within each Scrum Team before starting Scrum on a larger scale. We suggest scaling Scrum for experienced teams that have a good knowledge and understanding of values and workings of Scrum.



3. Scrum Artifacts

Three Scrum Artifacts are crucial for an effective Scrum Team: two Backlogs, which are lists of tasks, and the Increment, which is a potentially release-ready version of a Product improved in a given Sprint. They are collectively called Artifacts because they have one purpose. Namely, to maximize the transparency of information regarding the work on the Product. With the availability of Scrum Artifacts, any Scrum Team member or Stakeholder can get a clear picture at any time. There, they will see:

- What Product and for what Goal is being created?
- What tasks are planned to be performed?
- What tasks the Development Team is currently working on?
- What tasks have already been completed?
- What the current version of the running Product looks like?

Scrum backlogs describe the Product both from the technical and business angles. The technical description of the Product created by the Scrum Team contains the way the Product works, as well as proposals of specific solutions to be implemented by the Development Team. The business description contains User Stories that answer questions like: what is the Product for, what functions does the Product have to perform, what expectations of the Customer should the Product meet. Thus, they describe partial functionalities of the Product from the Customer's point of view. It is a good idea to link such documentation to the project, so that everyone involved can access it in real time. In Firmbee, you can upload files from your device or share a link, for example to a document in Google Drive.

1. Product Backlog

The Product Backlog is a list of tasks that the Scrum Team will work on. It is expressed in business language, and its scope covers the entire duration of the project. The maintenance and availability of the Product Backlog are crucial for the transparency of the Scrum Team's work. Thanks to this document, the Development Team knows what business problem they are working on solving and what the Customer's priorities are. What's more, the Product Backlog is a signpost on which you can support yourself when you get overwhelmed with smaller tasks that blur the picture of the whole project.

The Product Backlog keeps track of the Development Team's progress toward the Product Goal. It is managed by the Product Owner and should be updated regularly. So that at any time it gives a clear picture of the work that needs to be done. The tasks with the closest deadlines are described in the Product Backlog in the most detailed way. Tasks with longer deadlines or optional tasks have a form of a general outline.

2. Sprint Backlog

We can think of Sprint Backlog as similar to Product Backlog. However, the way tasks are described and the time scale changes. While in the Product Backlog the focus was on describing tasks from a Stakeholder and business language perspective, the Sprint Backlog is the domain of Developers. They are responsible for maintaining and updating the Sprint Backlog. **It is a description of the Team's work, therefore it is**

expressed in technical language. It describes detailed tasks and solutions as the Developers plan their execution. The Sprint Backlog operates on a time scale that corresponds to the duration of one Sprint, typically two weeks to one month. It allows for tracking of the Development Team's progress toward the Sprint Goal.

3. Increment

Product Backlog content is a step-by-step description of how to achieve the Product Goal. The content of the Sprint Backlog is a description of the tasks leading to the achievement of the Sprint Goal.

The Increment is the sum of partial Product functionalities realized in a given Sprint added to the current state of the Product.

Each new Increment builds on the previous one. Therefore, the work done should be thoroughly tested. Testing allows you to make sure that the new solution does not interfere with or disrupt those that were created earlier. **For the work to become an Increment, it must be integrated into the existing state of the Product** and result in its improved, working version. In other words, an Increment is a set of tasks completed in one Sprint that make up a new, working version of a Product. And its business meaning is described by the Definition of Done. It is entered into the Product Backlog during Sprint Planning.

Sprint Backlog

The Development Team creates a new Sprint Backlog during Sprint Planning. From that moment on, it becomes the current commitment for the Developers, i.e., a list of new functionalities, enhancements and modifications to the Product to implement in the starting Sprint. After the start of a Sprint, the Backlog becomes a binding queue from which Developers choose tasks to perform.

A Sprint Backlog describes the work of the Development Team during a single Sprint. Therefore, it is expressed in technical language. It describes detailed tasks and their planned solutions. Thus, it consists of a list of tasks drawn up in a way that is clear to Developers. The Sprint Backlog usually takes little account of the business value language of the Product, a way of description proper to the Product Backlog. The Sprint Backlog arises: based on the Product Backlog, for the duration of one Sprint, during a Scrum Event called Sprint Planning, by the entire Scrum Team – but the Development Team the key role in its creation.

How is the Sprint Backlog created?

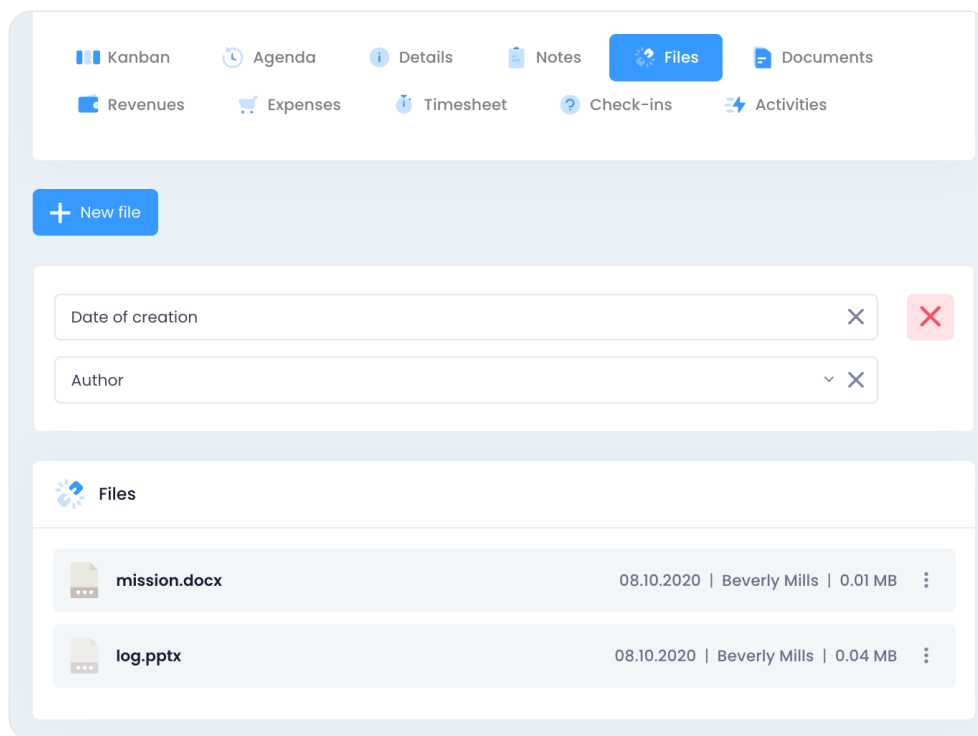
During Sprint Planning, the Product Owner proposes how to add value to the Product in the next Sprint.

Then the whole Scrum Team works together to formulate the Sprint Goal, i.e., selects which functionality from the Product Backlog to implement. The Sprint Goal defines how to implement the Product or postpone the deadline to meet the Customer's expectations.

The next step is to think over and set realistically the scope of work to do in the next Sprint and how to achieve it. The results of these findings come in the form of a technical description of the tasks to perform and this list becomes the new Sprint Backlog.

The newly created Sprint Backlog exists in a location that is easily accessible to all members of the Development Team. In the physical space, it is usually a whiteboard hanging in the workspace.

Whereas in the digital space, it exists as a cloud-based shared document that all Developers can update. Even though every member of a Scrum Team should keep it updated on daily basis, it's the Scrum Master or one of the Developers who usually take on that responsibility.



Source: Firmbee - Project documents

What does the Sprint Backlog contain?

The Product Backlog does not specify how exactly to execute tasks. It's the Development Team's role to decide. That move creates enough space for the team to maneuver thus enhancing its self-organization capabilities. Also, this freedom to select the sequence and methods of action empowers each Developer rendering a sense of independence and responsibility.

The same idea applies to treating the Sprint Backlog as an unordered list of tasks to execute. Contrary to the traditional push model (where the Team or Developer acts according to a predefined and imposed agenda), in the pull model, the Developers select which tasks to do (pull model). The Sprint Backlog specifies:

1. The Sprint Goal – i.e. an answer to the question of why to perform the scheduled tasks this Sprint
2. The list of new Product features and enhancements to develop in this Sprint. This is because it contains the Product Backlog elements selected for implementation in this Sprint.
3. The list of tasks to perform – that is, a technical description of how and by whom the work that will result in Incremental.

Using the Sprint Backlog

Various metric tools reflect the progress of work written in the Sprint Backlog. Most often it's the Burndown Chart. With such a visualization, the Development Team can easily see whether the work on the Sprint Goal is proceeding according to plan.

It may happen during a Sprint that you find that the work plan has been blueprinted unrealistically. In other words, the number of to-dos in the Product Backlog Sprint Goal is too high or too low. In either case, the Developers and the Product Owner get down to find out what changes to apply to the current Sprint Backlog. It is possible to reduce the amount of work, select additional tasks from the Product Backlog or extend the already planned solutions.



However, keep in mind that the Sprint Goal itself has to remain unaltered.

Product Backlog

Product Backlog is the largest of the Scrum Artifacts. **It reflects the status of work on a Product concerning the Product Goal.** On the other hand, when the work on a Product is completed, its Backlog becomes a complete list of the tasks done by the Scrum Team to create the Product. However, it does not contain detailed technical solutions. Product Backlog is created during the Product Owner's meetings with Stakeholders. The Product Owner is the sole owner and the person responsible for this source of tasks. Business language characterizes the entries in the Product Backlog. In other words, they describe the value of the Product from the Stakeholders' point of view.

Task descriptions included in the list of tasks need coherence and clarity. They contain functions and improvements of the Product usually presented in the form of User Stories. These are descriptions of partial functionalities of the product answering the questions about the following issues: the scope of Product modifications, the purpose of modifying the Product, the type of user for whom this modification comes up.

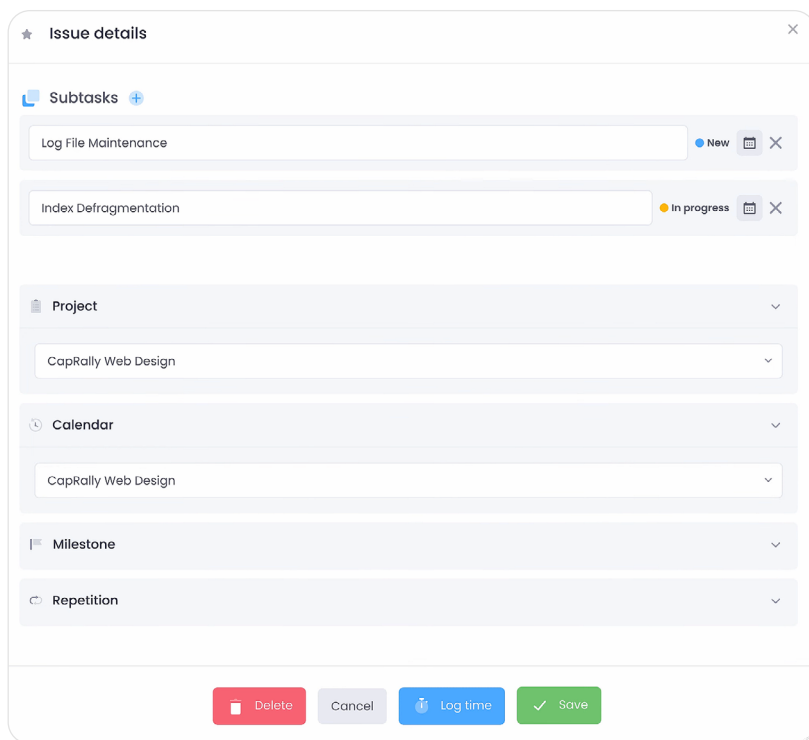
The shape of the Product Backlog

The order of tasks included in the Product Backlog changes as the Product develops. While working on it, the Scrum Team shapes and enhances its functionalities. Upon encountering obstacles, its implemented actions allow all to think about and define future adequate solutions, and these will too change accordingly to unforeseen further obstacle. Therefore, there is no clear and defined order of actions, all is changeable. Product Backlog improvement is aimed at its continuous updating and preparation for the next tasks. For this reason, it is continuous.

Tasks with a distant deadline are typically large, generic wholes. Their description does not contain details, but only an outline of functionality that should be realized. It is also possible to find tasks among them that will never terminate. Entries in the Product Backlog may present alternative solutions and also the Customer's ideas which may become outdated, unprofitable or for some other reason never enter the implementation phase. That is why the Product Backlog is sometimes jokingly called the "Customer's wish list".

Another reason for changes in the shape of the Product Backlog is redefining solutions. Sometimes it turns out that a certain problem has already been solved while creating another product functionality or the expected functionality has become redundant due to changes in other solutions.

One of the basic activities during the Product Backlog improvement is dividing the tasks contained in the Product Backlog into parts. Thanks to this, the general outline of functionality is presented in the form of smaller, more detailed and precisely defined units. In a project management system, each task can be divided into subtasks, and their status, priority and completion time can be specified.



[Source: Firmbee - Subtasks](#)

Tasks designed for closer implementation become more detailed. They also become smaller, containing details of solutions. Details emerge during Product development. And thanks to the knowledge of the current state of the Product and the current expectations of Stakeholders, the Product Owner complements the upcoming tasks with their description, order and size. Then, selects the best-described tasks for the next Sprint Backlog.

Product Backlog Improvement

While working on a Product, the Product Owner modifies and details the Product Backlog in cooperation with the Development Team. Following the Product Owner's suggestions, during Sprint Planning the team selects features to implement from the Product Backlog. They are then moved to the Sprint Backlog and divided into tasks to be completed. The tasks moved to the Sprint Backlog are described in technical language, which is the most useful for Developers. Task size is an important metric from the Development Team's point of view. Its proper estimation becomes especially critical when selecting User Stories from the Product Backlog to the Sprint Backlog. The Development Team learns over time to correctly estimate the time and effort required to complete a specific User Story. This is expressed in days, man-hours, or Story Points and provides an estimate of a value called Team Velocity.

Product Backlog nurturing

Product Backlog nurturing is one of the primary tasks of a Product Owner. The nurturing process includes formulating, detailing and adding new User Stories to the Product Backlog. However, the most important of the nurturing tasks is ensuring that the entries placed in the Backlog are in the right order, i.e. become prioritized.

Product Backlog is one of Scrum's Artifacts. It contains a prioritized list of work needed to create a Product. In other words, it is a list of User Stories necessary to achieve the Product Goal. Product Backlog nurturing also goes by the following names: Backlog Prioritization, Backlog Refinement, Backlog Scaling.

1. Goal of Product Backlog nurturing

The Product Owner manages the Product Backlog. The key skills include prioritizing tasks as their due date approaches. **This is because the goal of Product Backlog nurturing is to make sure that the Product functionalities come with the highest business value, i.e.** those most essential from the Customer's point of view, are at the top of the to-do list. And their description is clear and detailed so that their implementation can start right in the next Sprint.



The Product Backlog can get updated daily if needed. The Product Owner can add new User Stories to the Product Backlog after talking to Stakeholders and the Development Team, or by drawing conclusions and reformulating User Stories already written in the Product Backlog. **Mandatory updating of the Backlog is one of the tasks performed during Sprint Review.**

Usually, during this meeting, the Scrum Team discusses not only the tasks to complete in the next Sprint. It also preliminarily specifies User Stories and their implementation in the next two or three Sprints. This way of doing things allows the Scrum Team and its activities to take a broader view of the long-term direction. It enables to think of the tasks currently being performed from the perspective of their development in subsequent Sprints.

2. Errors in Product Backlog maintenance

One of the most common problems regarding the Product Backlog nurturing is allowing it to expand uncontrollably. This is because while working on the Product, various additional functionalities and tasks proposed by both Stakeholders and Scrum Team members spontaneously appear. Therefore, limiting the growth of the Product Backlog scope (scope creep) is one of the most important tasks performed by the Product Owner. The most common mistakes that Product Owners make concern:

- ◆ **Deviating from the Product Goal** – adding too many ideas to the Product Backlog beyond the basic Product Goal is not a good practice, as it greatly reduces its readability. It works better to collect ideas for additional functionality in a separate document.
- ◆ **Duplication of content** – entering repeated or very similar ideas from different Stakeholders into the Backlog – before adding another entry to the Backlog, the Product Owner should make sure that the new entry does not duplicate any of the existing ones.

- 🟡 **Lack of a broader perspective** – you should order Product Backlog entries according to their value concerning the Product Goal. Still, keep in mind that prioritization should take into account the next several Sprints so that the tasks performed in a given Sprint are seamlessly linked to both the preceding Sprint and the Sprint immediately following it.

You cannot avoid mistakes of this kind. However, awareness of their occurrence can make the Product Owner more cautious about adding new User Stories to the Product Backlog to work out the right balance. This is because it is also a mistake to give the Backlog too much cut and eliminate entries that contain similar tasks that differ. For example, describing similar Product functionalities that differ significantly in the application.

3. Backlog maintenance vs. metrics used in Scrum

The Product Backlog contains a description of the remaining work throughout the project. However, only an up-to-date and regularly nurtured Backlog can accurately estimate the ratio of the amount of work completed to the total. To depict the amount of work completed, you should apply the Burndown Chart. Another popular metric to describe Scrum Team work is Velocity. You can measure it by comparing the number of Product Backlog entries converted into Increment during a single Sprint.

User Stories

User Story is a brief description of a new Product functionality or its enhancement. It does not contain a technical solution but addresses questions concerning the functionality: Who is the user? What does the Product do? And What is its purpose? The User Story describes the product in everyday or business language, though it also points to Scrum Team's tasks that are intended to improve the Team's performance.

User Story is the most common way of formulating the tasks performed by the Scrum Team.

A single User Story defines a small functionality of the Product. It describes the smallest meaningful, partial Product Goal. For this reason, User Stories are very brief.

User Stories are created throughout the entire time of working on the Product. They are created continuously, from the moment the decision to start work is made, to the realization of the Product Goal. Creating User Stories is a Product Owner's task. Based on a conversation with a Customer, formulates answers to questions that allow to create User Story and enters them into the Product Backlog. However, User Stories reflect not only customer needs.

User-Story Map

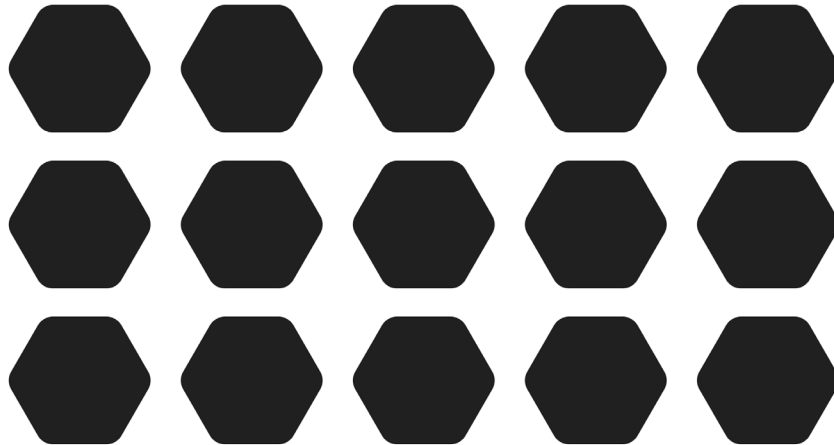
1. Activities:



2. Steps:



3. Details:



Whose story is it?

The Scrum Team creates a User Story to define the User's needs, and that's why it is put down in business language. In other words, it indicates the benefits that its implementation will bring to the product user. However, in the Product Backlog, there can also be User Stories that describe the needs of the Development Team, for example improving the workflow between Developers, or describing the needs of the Product Owner, for example organizing the Product Backlog. In such cases, the User in the User Story is the Developer and the Product Owner.

You can describe a User Story by answering the 3W questions:

- 🟡 **Who?** (is doing that),
- 🟡 **What?** (are they doing),
- 🟡 **Why?** (do they need it).

The User Story is then contained in a formula:

As a [user type], I want [to do what?] Because [why? why?].

Examples of User Stories about the functionality of an online store written in this form are illustrated in the table below:

	Who? What user?		What? What to do?		Why? What for?
As	Customer	I want	buy a magic wand with one click	since	I need it now.
	Developer		improve display of magic wand images		inscriptions are unreadable on smartphone screens.
	Product Owner		change the way User Stories are pinned to the board		fall.

This formula allows not only to formulate a User Story but also to relatively easily translate technical language into business and vice versa. As a result, both Developers and Stakeholders see clearly the Goal and stages of its progress.

How to use User Stories?

Creating a schematic User Story is just the beginning. They are signals and starting points for discussions on problems and their solutions. Discussing User Stories takes place during Sprint Planning to sort out which technical issues the Development team will add to the Sprint Backlog. Typically, in the physical space, User Stories are written on small, colored cards pinned in the workplace. However, in the digital space, digital whiteboards, shared by the Scrum Team, work best.

Saving User Stories in this way has several advantages because:

- **stresses the autonomy of each User Story** – each has a separate framework and can be executed independently of the others,
- **emphasizes the dynamics of User Stories** – the order of their realization is renegotiated by the Scrum Team and the current order of realization is visible on the board thanks to the physical arrangement of cards with User Stories,
- **serves as a reminder** – thanks to the visual representation of User Stories, the Scrum Team has a signpost in sight to remind them of the goal when creating detailed solutions.

The Development Team estimates the required effort to complete a User Story with days, man-hours, or Story Points.

Acceptance Criteria

A User Story must have certain Acceptance Criteria at the very moment it is accepted for development by the Development Team. **The Acceptance Criteria determine at what point work on a User Story can be considered finished.** This way both the client and the Developers know how their work will translate into business value. Typically, a User Story is considered completed when the user-specified in it can perform the action described. Using the example above, take a look at this User Story with the content:

“A customer can buy a magic wand with a single click.”

It is completed when a working “Buy Now” button appears on the online store page, which uses the default payment and shipping information for the logged-in user.

INVEST – creating the best User Story

INVEST is a method for creating good User Stories. It allows checking if they have properly formulated content and if they relate to the business value of the Product and also, whether their size and usability have been chosen properly. INVEST is an acronym created by Bill Wake in 2003. Each letter of it stands for the beginning of a word that characterizes a good User Story. According to the INVEST principle, every User Story should be:

1. I for Independent

The first feature of a good User Story is its independence. It means that its description and characteristics should be understandable without reference to other User Stories. But most of all, its realization shouldn't correlate with other User Stories. Of course, it will not be full independence. You can't divide Product creation into completely separate modules. However, it is crucial to remember about keeping User Stories as independent as possible. Thanks to that, even if one of them does not enter the implementation phase or is significantly modified, the remaining one will not have to be modified. As a rule, User Story should constitute a separate and coherent whole.

2. N for Negotiable

User Story should be negotiable. This means that it sets the Goal, not the way to get there. In other words, it defines an expected feature of the Product, not a technical solution to implement. User Story negotiation takes place between the Product Owner and the Development Team. The Product Owner proposes the implementation of certain functionality of the Product, i.e. says “What” to do. The Developers are responsible for answering the “How” question. That is, negotiating specific ways of solving the problem presented in the User Story.

3. V for Valuable or Vertical

In the acronym INVEST, the letter V stands for two qualities: valuable or vertical. Both reveal key characteristics of a good User Story. Therefore, we decided to explain what each of them means.

- A valuable User Story justifies the business purpose of the modification. In other words, it accurately answers the question of why to introduce the modification be introduced and why it is important from the Stakeholders' point of view.

- The vertical User Story contains a new feature of the Product visible to the User. That is, it does not focus on horizontal “performance improvement” in a selected layer of the Product. On the contrary, it adds another “layer” to it. In other words, User Story describes how to modify the overall operation of a Product by answering the question of What exactly to improve? It also means that each functionality of the Product builds on existing solutions. Vertical derives from the Agile methodology.

4. E for Estimable

A good User Story should be estimable, which means that it must clearly define the scope of modifications to make to the product for the User Story to be considered complete. This allows the Development Team to determine the time and effort required to complete it. The scope and difficulty of a task are usually estimated in units called Story Points. They are relative and each Development Team works out the Story Point value in practice based on previous experience.

5. S for Small

User Story accepted for realization by the Development Team has to be concise. That is, it should be no longer than one Sprint. If Developers discover during Sprint Planning that the User Story proposed by the Product Owner is too long, they should split it into possibly independent parts.

6. T for Testable

The last letter of the acronym INVEST stands for testable. **It means that the Product modification described in the User Story must hold water and be verifiable.** In other words, it should be possible to verify whether the solution implemented by Developers delivered the assumed value to a specific Stakeholder.

User Story mistakes

User Story can be a great tool for motivating the team to propose new solutions to problems presented from the user's perspective. But let's focus on what could go wrong when writing and using User Story.

1. Problems with 3W

A proper User Story answers the questions: Who? (Who is the target user of the Product?), What? (What capabilities have the Product, and what can it do?), Why? (What purpose does it serve?). However, problems may accompany the answers to each of these questions. The least common problem is the doubt about what should change in the product in response to the customer's needs. Therefore, we will focus on problems concerning Who? and Why?

Who? – user persona

One of the most common mistakes made when creating User Stories is not answering the question precisely enough: for whom? In other words, who is the user for whom the planned change is intended? Often a generic response pointing to the Customer or End User as the recipient of the change is not enough. The solution to this problem is to imagine the recipient as a specific persona. A persona is a model image of the target customer. In other words, a persona is a representation of the person who will use the Product in a specific way.

After analyzing your User Story, you may find that it tells the stories of different people at the same time. If there are many target users, it's worth considering splitting the User Story into smaller fragments to avoid contradictory, mutually exclusive, or simply ineffective actions.

🔵 Why? – a poorly defined goal

Sometimes the last section of the User Story becomes the source of problems. It should specify the business value of the changes made during User Story execution. Take a look at an example of User Story mistakes where the description of additional functionality replaces the goal:

“As a customer, I want to buy a magic wand with one click because I want to buy a flying carpet next week.”

Instead of giving the reason for buying the magic wand, this User Story adds another item to the potential customer’s shopping list. Therefore, when preparing a User Story, don’t forget about the reasons for alterations in the functionality of the Product.

2. Problems with 3C

We can break down the process of working with User Stories into three stages called the 3Cs:

- 🟡 **Card** – The card on which the User Story is saved,
- 🟡 **Conversation** – A conversation within the Scrum Team about the User Story card,
- 🟡 **Confirmation** – defining Acceptance Criteria confirming that a task has been completed.

🔵 Card

The memory card which stores the User Story has a limited capacity. Therefore, the most common problems concern the length and volume of the User Story. The User Story needs coherence and no beating around the bush, as they say, to such a precise degree that every word counts. This is because the problem of the User Story card has two dimensions. One is the way it is formulated: concise and containing a necessary minimum amount of enumeration. The second is the actual size of the User Story. One general sentence can express a huge number of tasks that cannot be completed during a single Sprint.

🔵 Conversation

The one-sentence formulation of the User Story is the starting point for a conversation with the Development Team. Therefore, **it is incorrect to treat it as a description of the task to perform.** It disables the possibility of negotiations and discussion on various ways of its implementation. User Story should not be treated as a description of requirements for new product functionality, it is rather an invitation to start a conversation about specific technical solutions that will lead to the realization of business value defined by User Story.

🔵 Confirmation

We wrote about the Acceptance Criteria that must be defined for each User Story in detail in the text describing what a User Story is. **One of the common mistakes, however, is the lack of vagueness of performance criteria.** A well-written User Story contains a description of the situation in which it is implemented. Its test is that the User takes advantage of the new functionality created by the Development Team. A useful tool for validating the User Story is to develop an acceptance test. This is usually on the other side of the card containing the User Story.

User Story Acceptance Criteria

User Story is a technique allowing businesses to deliver products and services meeting customer need to the maximum. User Story Acceptance Criteria enhance the assessment of new Product functionalities from the user's point of view. **The Acceptance Criteria should follow these guidelines: describe the new and improved functionality of the Product from the user's point of view and be unique for each User Story.**

The official Scrum Guide doesn't define User Story and its Acceptance Criteria. They are optional, but very common elements of Scrum work. Still, to ease our readers' curiosity we'll depict them as: *The conditions that a Product enhancement has to meet during a given Sprint to get approval from the User.*

1. How to formulate User Story Acceptance Criteria?

A well-written User Story contains a clear description of the context or situation it concerns, thus meeting the Acceptance Criteria. Still, it is just short sentence, too vague and ambiguous to straightforwardly pinpoint necessary considerations.

◆ Clarity and accessibility of Acceptance Criteria

Therefore, to prevent ambiguities, conduct and record a detailed conversation with the customer to determine the purpose of the implemented solution. **Remember that the final formulation of Acceptance Criteria belongs to the Product Owner.** Write them down together with User Story criteria before Sprint Planning. Each Scrum Team member has to read it and confirm that they understand and agree with the User Story Acceptance Criteria. Usually, the Acceptance Criteria are on the other side of the User Story card.

Properly formulated Acceptance Criteria allow the user to check if testing User Story follows its description. The criteria can take the form of a checklist with bullet points to tick when completed during the Product testing at the end of a Sprint. The matter is simple if the Product's operation is transparent to the User. However, the more complex the product, the more difficult it gets to test. Take complex software or large-scale services. Therefore, in most cases, a useful tool to validate User Story is to prepare an acceptance test.

◆ Acceptance test

If you decide to develop an acceptance test, put it down on the other side of the card containing the User Story. Later, the Scrum Team or an external QA team can carry it out. **The test must first and foremost contain a clear statement of whether the Product fails or passes the test.** It cannot contain percentage statements or intermediate evaluation.



If the User Story has more than one acceptance criterion, each requires separate testing. This way, it is much easier to determine which product functionality needs improvement or refinement and is especially important if new functionalities included in the User Story overlap or are independent of each other.

2. User Story Acceptance Criteria vs. Definition of Done

Definition of Done is an integral part of working in Scrum, which is the technical equivalent of Acceptance Criteria. However, you shouldn't confuse these two as they denote different commitments. **The Definition of Done is a clear and transparent description of the expected state of the Product after the completion of the Increment in the Product Backlog.** It describes the improvements made within the Increment. This stands in contrast with the acceptance criterion corresponding to User Story, which describes the Product functionality created during the last Sprint as it is perceived by the Customer.

For example, take this User Story with the content: *As a logged-in customer of an online store, I want to buy a magic wand with one click.*

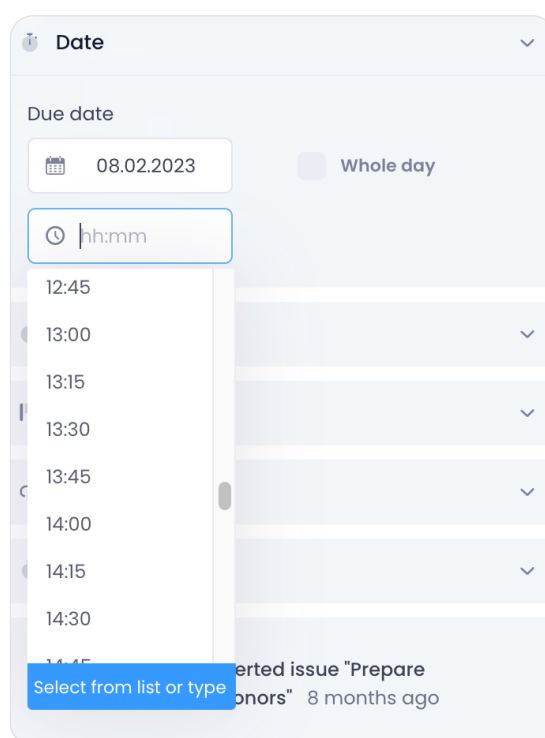
The Definition of Done for the above User Story might include the following: *the creation of a login panel for store customers, integration of the payment system, adding the instant payment button to the product page template*

On the other hand, the customer Acceptance Criteria feature: *the ability to log into the store, the possibility of defining a default payment method, working "Buy now" button for the "magic wand" product.*

Story Points and Estimation

Creating estimates in Scrum helps to predict the complexity and time required to complete tasks.

By analyzing the past, the entire Scrum Team collectively forecasts what the future holds. Therefore, the more experienced the Scrum Team is, the more accurate their estimates are. The team also collaborates on establishing the estimated time to complete the tasks during Sprint Planning, keeping in mind, that it's not a final commitment but a prediction. Its accuracy depends on numerous variables that constantly undergo unforeseeable changes and unexpected circumstances.



[Source: Firmbee - Deadline for task completion](#)

At each Sprint Planning, the Product Owner presents new User Stories to the team. The Product Owner selects them from the Product Backlog for implementation in the next Sprint. Then Scrum Team members jointly estimate the workload necessary to complete this new batch of duties. This kind of assignment is an estimation, requirements estimation.

It would seem that the simplest way is to define the time needed to complete the task in hours or days. However, practice and research conducted since the 1940s prove otherwise. Humans are unable to accurately estimate the time required to complete even very well-defined tasks. Besides, the number of hours needed to complete a task depends on who is doing the task, and what has – or has not – been done before. This is why Scrum typically uses units called Story Points.

If the Scrum Team has already worked on a similar project, knowing how much time certain tasks took will allow them to better estimate the deadline for the next challenges. In Firmbee, each user can log their working time, so they know which tasks they are able to complete faster and which ones take them more time.

The Importance of Story Points in Scrum

Each Development Team puts into practice the value of a Story Point by drawing from experience and the size of individual tasks, i.e., following the principle of empiricism. Most often, during Sprint Planning, the Scrum Master selects one or more samples of completed User Stories, which serve as a reference point for determining the value of the User Stories to develop.

That's why you can't assign values in Story Points without the context. For example, if the first task is assigned a value of 10, subsequent tasks will be evaluated against it as either greater or lesser. In this way, **within a Scrum Team project, all the tasks in the Product Backlog are related to each other.** This means that similar tasks performed by one Development Team will receive a similar number of points.

Story Points are relative units. This means that:

1. The Story Point value relates only to the tasks performed by a particular Scrum Team. Story Points describe the speed of completion of the tasks of one team. In other words, a User Story estimated at 10 Story Points by Team A, can get 50 by Team B. This is because, as we mentioned, their value is relatively calculated to other tasks performed by that team, and their experience with similar tasks.

2. The value of Story Points completed in one Sprint cannot be the basis for comparing the performance of two Scrum Teams. In order to avoid mistakes in managing Scrum projects, it is important to remember that the Speed of a Development Team expressed in Story Points done in one Sprint cannot be used to compare the performance of two Teams. This is because they could do the same work in parallel Sprints, which one Team estimated at 10 and the other at 50 Story Points.

It should also not be forgotten that the estimation contains many unknown elements and is made on the basis of incomplete data. For this reason, the predictions of even a very experienced Scrum Team can sometimes be very different from the real effort needed to complete a User Story.

Relative estimation techniques

What are the most effective estimation techniques in Scrum? There is no one-size-fits-all method that works for each team. Among the estimation techniques within agile methodologies, the most common are:

- **Planning Poker.** This most popular relative method takes a card game to calculate the amount of work to complete a task.
- **Team Estimation Game.** This one involves assigning execution of User Stories in a given Sprint with appropriate numerical values selected from the Fibonacci sequence.

Scrum, on the other hand, rejects the classic Absolute Estimation way of the traditional project management methodology. The way it estimates tasks is by defining in advance the person-months, duration, and cost of the whole project. This is a long process, difficult to implement, and requires the participation of experts who tend to establish the rationale and the code of conduct, but take no action who will not necessarily perform the tasks whose value they estimated. In other words, it is not only tedious but also highly inefficient.

Planning Poker

Planning Poker is one of the most popular Scrum estimation techniques. It takes place during Sprint Planning and has a few simple rules. All players become members of the Development Team and each of them simultaneously puts on the table a card with the number of Story Points to estimate the task described by the Product Owner. What are the advantages and disadvantages of Planning Poker, and how to play it?

Planning Poker, also called Scrum Poker or Pointing Poker is a relative technique for estimating the amount of work necessary to complete a specific task. It was created in 2002 by James Grenning. He wanted to solve the problem of endless disputes in the Scrum Team concerning the estimation of the difficulty of the tasks given to the Developers.

1. How to play Planning Poker?

The goal of Planning Poker is to estimate the difficulty and effort of each User Story selected for a given Sprint. The rules of playing Planning Poker are simple. However, first, you need to prepare the necessary accessories.

Planning Poker game prompts: a deck of cards with User Stories – prepared separately for each game and a deck of cards with Story Points – one deck for each Developer, for repeated use.

The Story Point cards usually contain values corresponding to the Fibonacci sequence, i.e. 0, 1, 3, 5, 8, 13, 20, 40, and 100. It also happens that they are marked with successive powers of 2, i.e. 2, 4, 8, 16, 32, and so on. Why aren't they consecutive numbers? Because Planning Poker is all about clearly showing the differences between the difficulty of the tasks. And too small differences between card values would obscure the judgments. Numbers usually express the number of Story Points. However, they can also be other units of measurement used by the Scrum Team.

The phases of Planning Poker:

1. User Story presentation	
2. Discussion	Phases 2 and 3 are repeated until reaching consensus by all
3. Gameplay	
4. Consensus	
5. Moving on to the next User Story	

Planning Poker usually takes place during Sprint Planning. The Product Owner holds the User Stories cards and the Developers receive a deck of cards with Story Points. The moderator is the Product Owner who starts the game by presenting one User Story to the other Scrum Team members. If they have questions, they should speak out immediately after presenting a User Story.

The next step is to start a discussion about the implementation of the User Story. The whole Scrum Team participates in the discussion, but the main participants are the Developers. The discussion concerns, among others, such issues as:

- technical side of the task,
- skills of individual Developers that will be necessary to complete the task,
- ways of dealing with the expected difficulties,
- additional tasks connected with User Story execution.

When the Developers agree on the most important issues, they each choose one of the cards from their Story Point deck. Then, they place it accordingly to their opinion over the User Story card that best reflects its level of complexity. The next step depends on how the cards got distributed:

- **If the Developers placed cards of different values on the table, they return to the discussion.** Then they take the cards off the table and re-estimate the value of the User Story. The situation repeats and the Developers draw again until they reach a consensus.
- **If the Developers agree on the User Story, they move on to the next round of Planning Poker.** The Product Owner presents the next User Story, and the procedure repeats until the pool of User Stories planned for the current Sprint is exhausted.

2. Advantages and disadvantages of Planning Poker

The advantage of Planning Poker is undoubtedly the standardization of work with User Stories. The Development Team holds in their hands a ready-made set of cards for calculating the amount of work. This enables the values in every Sprint to remain constant and the Team learns to estimate with specific units. Another important advantage is the equal participation of all Developers in estimating the task complexity. Even people who are not directly involved in its execution can contribute to the discussion. For example, by drawing attention to problems that did not occur because, for instance, Developers focused on the technical aspects of the task.

Another benefit of playing Planning Poker concerns developing the skill setting time limits on the discussion and, if necessary, limiting the number of rounds played for each User Story. However, the time required to reach consensus is also one of the most frequently cited drawbacks of Planning Poker. If one or more Developers are unwilling to agree with the others, the game can potentially drag on indefinitely.

Team Estimation Game

Team Estimation Game is a technique facilitating Sprint Planning in Scrum. It is also called Swimlanes Estimation. The latter term originated as spontaneous observation of a card game as the display of cards resembled the swimming lanes of a water pool. Team Estimation Game is constantly gaining popularity, as it enables Development Teams to create estimates about 3 times faster than using Planning Poker.

1. Rules of Team Estimation Game

Team Estimation Game prompts: a deck of User Stories cards – prepared separately for each game and a deck of Story Point cards – for repeated use.

First, stack the User Stories cards in the order corresponding to the entries in the Product Backlog. To ensure the most urgent ones get estimated first. The scoring cards usually contain values corresponding to the Fibonacci sequence. This is a sequence of the following numbers: 0, 1, 3, 5, 8, 13, 20, 40, and 100. You can also label them with successive powers of the number 2, that is, 2, 4, 8, 16, 32, and so on.

To play the Team Estimation Game, the Scrum Team members sit around a table. The phases of Team Estimation game:

- ◆ **Product Owner begins** by drawing the first card from the User Story deck and sharing its content with all. Then, the cards stay on the table. Then Product Owner explains to the rest of the Scrum Team that from now on, players will evaluate User Stories as easy or hard to implement by placing them accordingly to the left and the right. If any happens to have some degree of difficulty, the player will stack together, one on top of the other on the table. Now, the person sitting next to them clockwise makes the next move.
- ◆ **A player draws a card from the User Story deck.** After sharing its content with all, explains its essence to the Product Owner. The person holding the card then places it on the table and selects a seat based on their opinion of the difficulty of this User's Story. Then, the player explains the rationale behind the choice to all and the other player are free to ask questions concerning the reasoning. They can't question the decision itself but the arguments justifying the decision.
- ◆ **Now, players take a turn and have two options to choose from:** repeat step 2, or move one of the cards on the table to its most appropriate position. If they go for the second option, they should also justify what made them change their mind. Players take turns repeating step 3 till there are all cards from the User Story deck get distributed and estimated.
- ◆ **The final stage of placing the User Story cards** happens once, or many times, depending on Scrum Team practice. During this round, each player has yet another opportunity to move one of the cards on the table to a more appropriate place.

- ◆ **Once the players assign all the User Stories cards** to their locations representing levels of difficulty, the Development Team moves on to match value by assigning the cards from the Story Point pile. The first User Story card on the left gets the Story Point card with the lowest number of points by the Product Owner. The rule for placing subsequent cards is the same as for points 3 and 4. This completes the estimation.

2. Team Estimation Game vs. Planning Poker

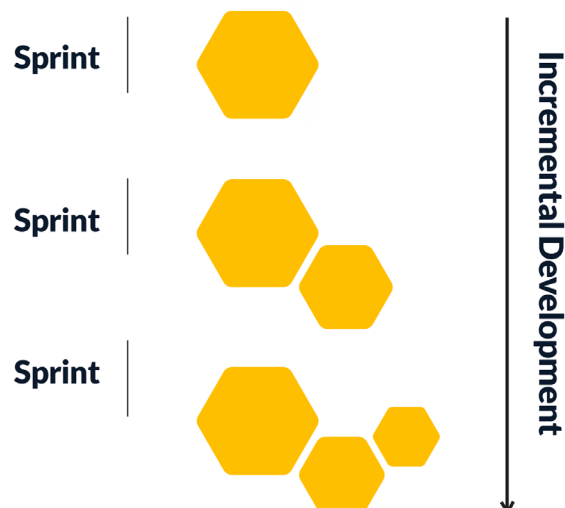
The Team Estimation Game is considered a more effective estimation tool than Planning Poker. Because of the following differences between these two techniques:

- ◆ **Card-table.** The Team Estimation Game uses the well-known “card-table rule” from popular card games. This means that once you have placed a card, you cannot take it back. Because User Story gets estimated by one person at a time, the fluctuation between estimations and the number of times the shift positions is significantly lower, in comparison to Planning Poker.
- ◆ **An accurate enough calculation.** In Planning Poker, a full consensus should be reached for each User Story. In Team Estimation Game, however, only one person decides. Even if his/her estimation is wrong, another Developer will likely place it on matching its value more precisely. This way guarantees reaching a sufficiently accurate and quick estimates
- ◆ **Exhausting the subject of discussion.** Arguing choices often get excessively long when playing Planning Poker. Their time is greatly reduced during a Team Estimation Game because they focus on a single decision by one of the Developers rather than the nature of each User Story.

One potential drawback of the Team Estimation Game is a sense of unfairness. If the Development Team is larger than the number of User Stories scheduled in a given Sprint, some Developers may feel left out.

Increment

An Increment is the latest completed and improved version of a Product that has business value and is ready for release during a Sprint. An important part of its definition is the statement that this is the sum of the previous state of the Product together with its amendments added in the current Sprint. Thus, it does not define only the new functionalities supplemented to the Product, but also – or primarily – their integration with its original version together with all the improvements and fixes it has undergone.



Increment vs. Product Goal and Sprint Goal

Each Sprint should produce at least one new Increment – it is the responsibility of the entire Scrum Team. So how does it relate to the Sprint Goal? **You could say that Sprint Goal is the answer to why we should construct one.** However, keep in mind that in a single Sprint the team can create several Increments, the sum of which combine into a Sprint Goal.

Thus, an example Sprint Goal might be the following task: *Create a “shopping cart” functionality in an online magic accessories store.*

However, the components of the Increment will include several functionalities of the store such as: *adding an item to the cart, removing an item from the cart, displaying sums due for purchases, calculation of delivery costs for items in the basket, etc.*

The team needs to envision that each Increment is a step forwards toward achieving the Product Goal. Therefore, to enhance the process of its improvement, it can: *order the customer database, augment the workflow between Developers.* That's because although they are not directly related to improving the current state of the Product, they have a huge impact on better and swifter realization of the Product Goal.

When does the work done become an Increment?

The measure of a successful Increment is whether it is practical. In other words, **the work done becomes an Increment only when it meets the Definition of Done.** This definition uniquely states what changes and improvements have occurred to the Product. Based on the Definition of Done you can test the Increment to assess if it really introduces improvements to the Product and if it brings the claimed business value. If the result does not meet the standards of Definition of Done, the project goes back to the Product Backlog. What comes next can take the following paths:

- ◆ the team may finish it during the next Sprint,
- ◆ the project may be canceled,
- ◆ the team may decide to keep it in the Product Backlog for future execution – once the team works out the way to overcome the obstacles.

If the team manages to produce an Increment in a Sprint, it moves forth to get evaluated during Sprint Review. However, if necessary, it may be shown to Stakeholders even before the end of Sprint. The ultimate decision to release it belongs to the Product Owner.

4. Scrum Events

Scrum Events are meetings or organizational activities undertaken by the Scrum Team. Their common characteristic is that they serve the purpose of nurturing transparency: they are opportunities to talk about planning, methods and reflect on Scrum Team achievements. However, each Scrum Event has distinct features, a defined purpose and duration. Scrum Events, unofficially called “Scrum ceremonies” are: Sprint, Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. Each of these has a very specific role in Scrum.

1. Sprint

Sprints determine the work cycle of the Scrum Team, therefore each Sprint has the same length. They follow one another consecutively and last from the first day of work on a Product to the last. The Scrum Team establishes the duration of one Sprint. It depends on the needs and capabilities of the team and the organization, as well as on the nature of the Product that the Scrum Team is working on. The optimal Sprint duration should follow the principles of empiricism, according to which it should be:

- **long enough** – so that the Development Team has time to significantly improve the Product,
- **short enough** – to reduce risk, unnecessary complexity increase and Goal dilution.

Best practices suggest that one Sprint should last from one to four weeks. Shorter Sprints help you diagnose issues and constraints faster and estimate the amount of work to do in a given Sprint. Longer Sprints allow you to evaluate the results of the Team’s performance by presenting larger deliverables during a Sprint Review. During each Sprint all the smaller Scrum Events take place.

2. Sprint Planning

Sprint Planning, as the name suggests, is the Scrum Team meeting that concerns scheduling the next Sprint. Therefore it is held on the first day of a new Sprint. The entire Scrum Team participates in Sprint Planning, and the event lasts a maximum of eight hours. In a nutshell, it involves answering three questions: what will be the Goal of the Sprint, what to do, how to do it?

3. Daily Scrum

Daily Scrum is a short event in which the Developers participate. This daily meeting lasts no more than 15 minutes. To keep it simple, the Daily Scrum is always held at the same place and time. The main focus of the Daily Scrum is planning, not summarizing completed tasks. Therefore, the conversation about the current progress of the Product should focus strictly on Sprint Goal. The Development Team decides collectively on the upcoming scope of activities that each Developer should complete by the next Daily Scrum.

4. Sprint Review

Sprint Review, along with Sprint Retrospective, is a recap event and takes place on the last day of the ending Sprint. The purpose of the Sprint Review is to summarize the Sprint in terms of completed tasks

that aim at completing the Product Goal. Usually, it also includes a presentation of the current state of the Product and a discussion of its newly completed features. Each Sprint Review should end with an update of the Product Backlog. A Sprint Review is an open event: the entire Scrum Team as well as free-will Stakeholders, attend it. The event is usually moderated by a Scrum Master.

5. Sprint Retrospective

The Sprint Retrospective, also a Sprint summary event, is different from the Sprint Review. It focuses on how the Scrum Team is working, rather than on the product they are developing. To create a safe space for communication, Sprint Retrospective is an event involving only the Scrum Team members and Scrum Master chairs the session. The Sprint Retrospective lasts a maximum of three hours and is the last event of the ending Sprint. Its goals are: learn from the current collaboration methods, identify areas for improvement suggest and discuss how to make improvements.

The screenshot shows a web-based form for creating a meeting. The form is titled "Issue details" and has a close button (X) in the top right corner. At the top, there are radio buttons for "event", "meeting" (which is selected), and "task". To the right of these are a green plus icon, a "Generate To do" button, a "New" button, and a clock icon. Below this is a text input field containing "Sprint planning".

On the right side, there are several dropdown menus: "Project", "Calendar", "Scrum team", "Date", "Estimated hours", "Milestone", and "Repetition". The "Date" section includes "Start" and "End" date pickers, both set to "03.02.2023", and time pickers for "8:00" and "15:00". There is also a "Whole day" checkbox.

The main content area includes a "Guiding Questions" section with three prompts: "What will be the Goal of the Sprint?", "What will be done?", and "How will it be done?". Below this is a "Characters left: 9905" indicator and a "Room nr 5" input field. There are sections for "Online Meeting" (with an "Add online meeting" dropdown), "Assigned user" (with a user selection icon), "Contact" (with a "Choose" dropdown), "Attachments" (with a plus icon), and "Comments" (with an "Add comment" input field and a green checkmark button).

At the bottom of the form are "Cancel" and "Save" buttons.

Source: Firmbee - Planning of meetings

Sprint in Scrum

Several smaller events make up a Sprint in Scrum. Sprints, in turn, form together a path aimed at developing and releasing a Product. Each Sprint has a specific Sprint Goal and a Sprint Backlog maintained by the Development Team. Sprints are the largest of the Events in Scrum. They follow a continuous cycle from the beginning to the end of work on a Product and each iteration brings the team closer to achieving the Product Goal.

Each Sprint has a specific Sprint Goal to ensure consistency in the work of the Development Team. It takes the form of a business goal and answers the question “Why?”, “To what end?”, or “Why?”. **The workflow of a Sprint is documented in the Sprint Backlog, which lists the work needed to achieve the Sprint Goal.**

Sprint Structure

Each Sprint has a specific structure and includes the following events:

- **Sprint Planning** – the Sprint starts. During this event, the Scrum Team selects the planned work from the Product Backlog to be done in the new Sprint.
- **Daily Scrum** – a daily event where Developers plan the tasks for the day.
- **Sprint Review** – open to Stakeholders, held on the last day of a Sprint. Its purpose is to summarize the Sprint in terms of progress on the Product.
- **Sprint Retrospective** – the closing event of a Sprint, where the Scrum Team discusses the ways of working and ideas for improvement.

The repetition of Sprint events promotes the implementation of good organizational practices. In other words, the Scrum Team implements the routines necessary for effective planning and, while working, draws attention to problems that can be discussed at appropriate events.

Sprints and the three pillars of empiricism

Sprints render the Scrum Team to break down the work on the Product into equal time segments lasting no more than a month. This fixed framework reinforces the three pillars of empiricism: transparency, inspection and adaptation. Let's take a look at how they apply to Sprint and its structure.

1. Transparency

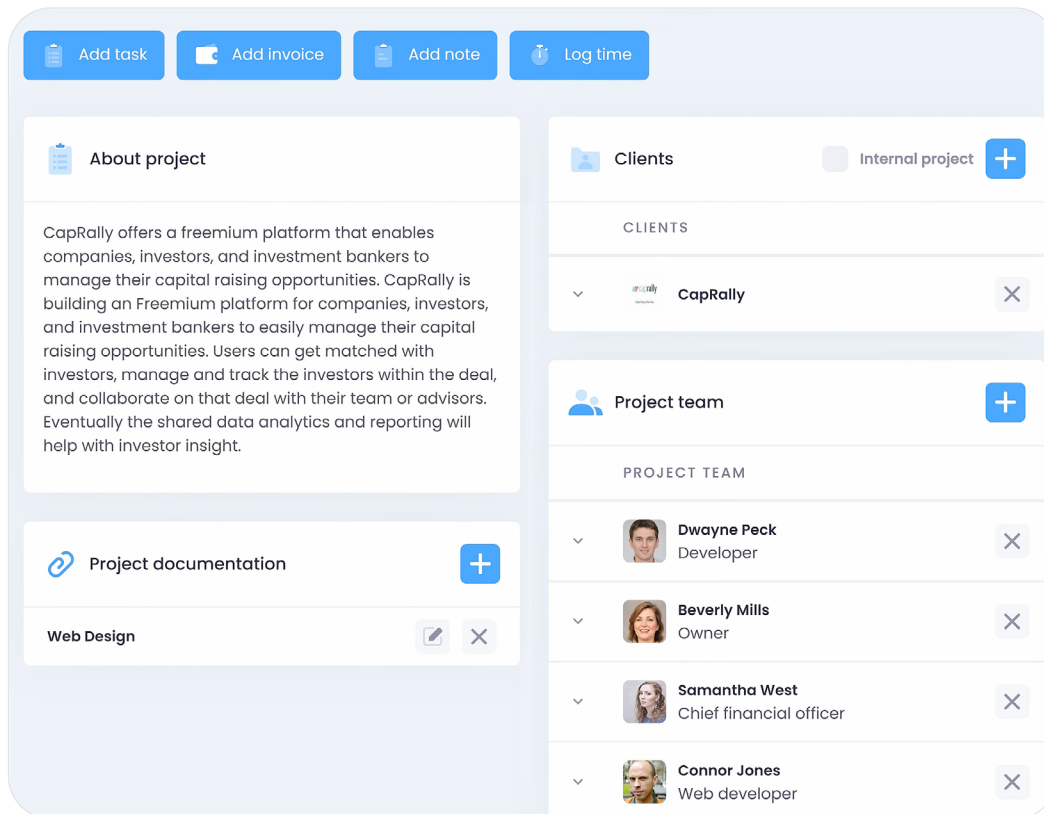
Splitting the work into Sprints enhances transparency as all the people involved can get the required information about the status of the Product work in each Sprint. Sprint Planning and Sprint Review, the beginning and the end of a Sprint, combined with an update of the Product Backlog, provide all Stakeholders with valuable insights into the current status of the Product.

2. Inspection

By dividing the work into Sprints it is possible to frequently monitor its progress. This promotes the constant identification of problems in two key areas. These are:

- **issues related to the achievement of the Product Goal** – at the start and end of the Sprint, i.e. during Sprint Planning and Sprint Review,
- **obstacles in the way the Scrum Team works** – during daily meetings and at the end of each Sprint, i.e. during Daily Scrum and Sprint Retrospective.

Dedicated project management tools help maintain transparency. Team members can monitor their work progress in real time, view the shared documents or control the budget.



Source: Firmbee - Project planning

3. Adaptation

Adaptation is a very important part of the Scrum Team's work, as it allows to solve the problems identified during the inspection. During each Sprint, Daily Scrum and Sprint Retrospective provide a safe space to talk about how to improve the Scrum Team. Implementing proposed solutions happens immediately or at the start of the next Sprint.

Sprint Planning and Sprint Review create a safe space for discussion concerning the Goals and methods to achieve them. A good self-managing Scrum Team successfully figures out what and how to implement for the next Sprint.

What changes to make during a Sprint?

Every Sprint leaves enough space for Scrum Team to improve and improvise the way they work.

Therefore, identify what to change during a Sprint. The Scrum Guide does not provide a list of such changes. However, the notion of empiricism provides guidelines to follow and adapt to the way a particular Scrum Team works.

1. All changes can jeopardize the achievement of the Sprint Goal. According to the first rule, during a Sprint you cannot, for example, reduce the number of to-dos in that Sprint, or significantly change their characteristics. A Sprint is closely tied to the Sprint Goal. Therefore, when the Goal changes, we should abort the Sprint. However, this hardly ever happens as the only reason for a Sprint to fail is when the Goal becomes obsolete. Keep in mind that the decision to terminate a Sprint belongs solely to the Product Owner.

2. The quality of work cannot be compromised. This rule is intended to prevent work done during a Sprint from becoming an Increment because it does not meet the Definition of Done. Reducing the quality of work can result in the Sprint Goal being ostensibly met, but the way individual tasks get completed does not meet the quality standards set by the organization or required by Stakeholders.

3. The Product Backlog can become detailed. While working on a Product, the knowledge about it increases. Therefore the detail of tasks to perform increases naturally. Therefore, it is an acceptable and even advisable change during a Sprint to detail the Product Backlog.

4. The scope of work may get clarified or renegotiated. This change, like the previous one, involves a growing understanding of the nature of the work being performed. The Development Team can do it in consultation with the Product Owner. However, the basic condition for its introduction is the lack of conflict with principles 1. and 2.

Product Goal, Sprint Goal and Definition of Done

Each Scrum Artifact creates a certain Scrum Team commitment. **The Product Goal, Sprint Goal, and Definition of Done describe the future state of the Product.** They are the intentions or commitments of the Scrum Team. However, each Scrum Team commitment covers a different scope and time scale. The Product Goal was not added to the official Scrum Guide until its 2020 version. It is defined there as a Scrum Team commitment to the final shape of the Product. It is made by the Scrum Team and written in the Product Backlog.

The Product Goal thus joined the previously existing Goals in Scrum: The Sprint Goal, which is about the Sprint Backlog, and the Definition of Done, which is the Scrum Team's commitment to delivering the Increment according to agreed-upon guidelines. The creators of Scrum thus decided to make the following connections:

Commitment	Scrum Artifact
Product Goal	Product Backlog
Sprint Goal	Sprint Backlog
Definition of Done	Increment

Committing is designed to strengthen the Scrum Team's focus. It provides the team with a clear goal and horizon for the entire project, one Sprint, and an Increment. They should all keep in mind the Product Vision and the long-term strategic goal. The Vision is not a commitment of the Scrum Team, but rather the horizon of all its aspirations. It helps in formulating the Goals accordingly. It can be formulated as follows: *To become the world leader in magic commerce and services.* In the text below, examples of sub-goals will refer to this example of the Product Vision.

1. Product Goal

The Product Goal is a description of the future Product that the Scrum Team is working on from the beginning to the end of the project. The path to achieving the Product Goal is written in the Product Backlog. A Scrum Team can only do one Product Goal at a time. To change it, the Scrum Team must either complete it or abandon it. A good formulation of the Product Goal depends on the industry in which the Scrum Team works, but in all cases, the Goal should be compelling, clearly expressed, and relevant. In other words, it should describe the key achievement that brings the Product Vision closer to realization. It could be, for example:

Product Goal 1: To open the world's first online store for magic accessories.

Product Goal 2: Create a 24/7 flying broom service.

2. Sprint Goal

A Sprint Goal is a description of the work that will be done within a Sprint. It is expressed in the form of a Business Goal to ensure consistency in the work of the Development Team, and also increase focus – one of the core values of Scrum. The completion of Product Goal 1 that we proposed above could be broken down into the following Sprint Goals:

Sprint Goal 1.1: *Conduct a market survey of magicians for demand for magic accessories.*

Sprint Goal 1.2: *Compile a list of suppliers and product availability.*

Sprint Goal 1.3: *To create a store website.*

Whereas for Product Goal 2, the Sprint Goals could look like the following:

Sprint Goal 2.1: *Find a convenient location for the first service point.*

Sprint Goal 2.2: *Conduct service technician recruitment.*

Sprint Goal 2.3: *Launch a local marketing campaign.*

Each Sprint Goal is defined and discussed by the Scrum Team during Sprint Planning. A Sprint Planning event cannot be completed without defining the Goal for the next Sprint.

3. Definition of Done

The Definition of Done is another detailed commitment made by the Scrum Team. Unlike Product and Sprint Goals, it is formal by definition. It usually consists of a list of quality criteria that must be met by the Product for the Increment to be considered finished. Thanks to the Definition of Done, Scrum Team members and each Stakeholder can easily see what changes and improvements have been made to the Product by a given Increment.

For Sprint Goal 1.1: *Survey the mage market for demand for magic accessories*, a criterion for the Increment could be: *Compile survey results of mages active in the magic forum*. The Definition of Done this increment could be as follows:

Definition of Done Increment 1.1:

- ◆ *Each of the magicians gave comprehensive answers to the questions.*
- ◆ *At least 80% of magicians are active.*
- ◆ *The results of the study were entered into a magic bullet database.*

The Definition of Done can also refer to quality standards defined by the organization. Then it is not freely set by the Scrum Team. Its minimum requirements are set by the external requirements of the organization. However, the Definition of Done differs from the Acceptance Criteria. Referring to the example of Definition of Done 1.1, the Acceptance Criteria could be formulated as follows:

Acceptance Criteria 1.1:

- ◆ *Estimate the percentage of magicians who will be willing to use an online store.*
- ◆ *Identify which accessories are most popular among them.*
- ◆ *Determine what budget your potential customers have.*

The Acceptance Criteria, therefore, refers to the performance of the task as it is perceived by the Customer.

Burndown Chart

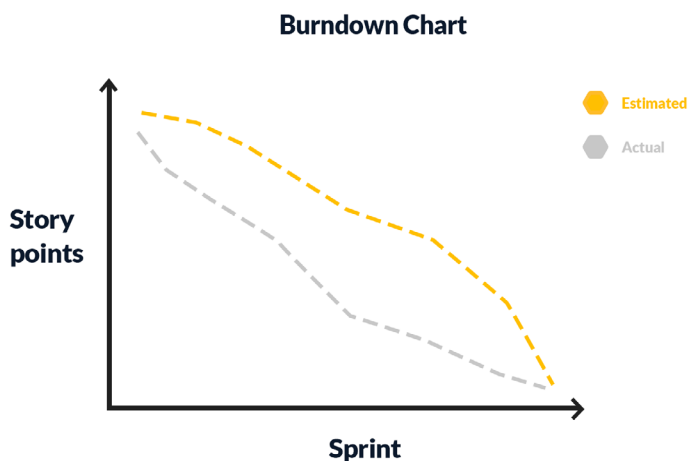
The Burndown Chart allows you to assess the stage of completion of the current Sprint or the entire project being worked on by the Scrum Team. But most importantly, it shows how much time is needed to achieve the Sprint Goal or Product Goal. To some extent, it also enables you to measure changes in the effectiveness of the Development Team and thanks to that, it is possible to assess the probability of finishing the tasks on time. This is because the Burndown Chart shows how much work has already been done and how much remains.

A Burndown Chart serves to estimate the time and effort needed to complete a task. It is also a valuable warning tool if the work does not go as planned. With a single glance, each Scrum Team member is able to assess the status of the tasks. Therefore, the Burndown Chart should be easily accessible to everyone involved in the project. The Burndown Chart in Scrum features in two versions: as a Sprint Burndown Chart and as a Project Burndown Chart. We will discuss them together because the principle behind them is identical. They differ only in the time scale and the number of tasks they depict.

What does a Burndown Chart look like?

On the Burndown Chart, the X-axis shows the time remaining to complete the work. If the chart is for a Sprint, the time is measured in days. On the other hand, if the Burndown Chart is for an entire project, time is usually measured in Sprints.

The Y-axis of the Burndown Chart indicates the amount of work remaining. Thus, it refers to the tasks included in the Sprint Backlog or the Product Backlog. The units used on the vertical axis of the chart can be: Story Points, Man Days/Hours, Tasks. Burndown Charts typically includes a burn-down line that represents a perfect, linear decrease in the number of tasks that remain to be completed. It provides an outlook on the projected completion time.



Estimation of task completion time

Each Developer individually needs to estimate the number of hours they need to complete a particular task and the estimated time to complete a Sprint Goal or Project is the sum of time estimated by all Developers.

The immense difficulty is to translate the estimated time into the actual time of task execution. This is one of the major problems newly formed Development Teams face. It can distort the picture presented on the Burndown Chart.

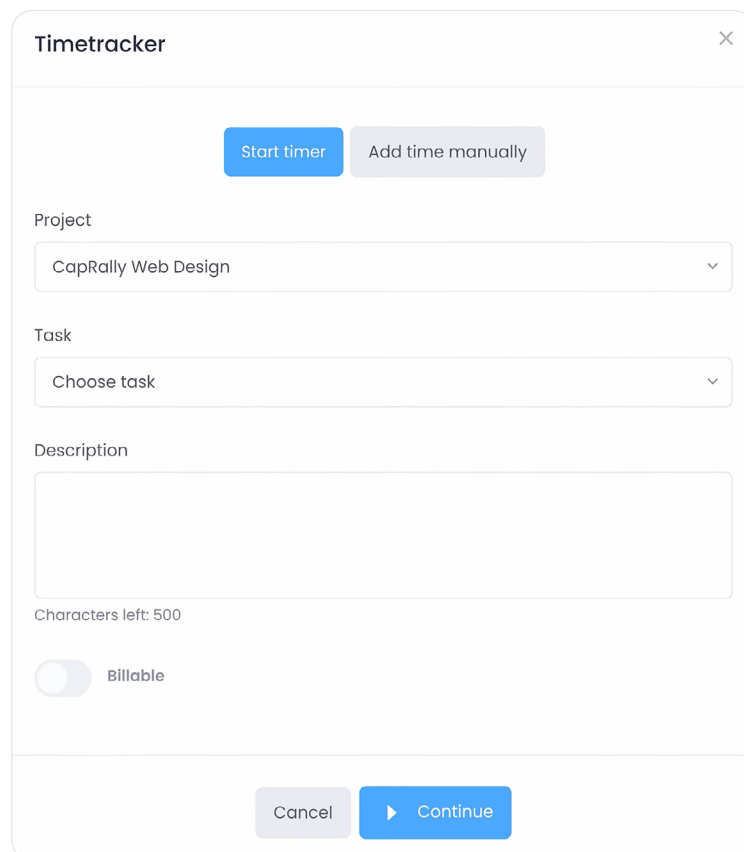
Moreover, it is not always possible to create an optimal workflow in a Development Team. As a result, task execution may suffer from downtime, which increases the task completion time. One of the indicators of Team maturity is the ability to accurately estimate the work time. The more predictable, the more mature the team is. This is reflected in the burn rate graph, where the line reflecting real burn rate descends steadily to zero. However, this formulation does not apply to all situations.

The statistics monitored by Scrum Master, much depends on the nature of the project itself: its innovation and the repetitiveness of the tasks performed by the Developers. The Burndown Chart works best for projects where the scope of work is fixed and known. Therefore, in case of innovative projects it is better to use other ways of measuring team effectiveness.

How to create and interpret a Burndown Chart?

A Burndown Chart is relatively easy to create. There are many tools available to generate it from the work logged by members of the Development Team. Despite its simplicity, its interpretation can provide valuable information for the entire Scrum Team.

The Development Team should monitor its daily work. This is the basis not only for evaluating its effectiveness but also for improving it and one of the simplest and proven tools for this purpose is the Burndown Chart. You can create it manually by drawing a coordinate system on a piece of paper. On the Y-axis you need to plot the amount of work expressed in a chosen unit, for example, story points. On the X-axis, draw a scale indicating the consecutive days of the Sprint. Draw a line of the ideal sprint then mark the number of realistically completed tasks for each day. Although this solution is charming and engages the team, it is not very practical. It is also not necessarily suitable for remote teams. Remote teams, however, can take advantage of the time logging feature that is available in the project management system. The time spent on a task can be logged by turning the timer on and off, or by entering the time manually.



The screenshot shows a 'Timetracker' window with a close button (X) in the top right corner. At the top, there are two buttons: 'Start timer' (blue) and 'Add time manually' (grey). Below these are two dropdown menus: 'Project' with 'CapRally Web Design' selected, and 'Task' with 'Choose task' selected. A large text input field for 'Description' is below the task dropdown, with a character count 'Characters left: 500' underneath it. A 'Billable' toggle switch is located below the description field. At the bottom of the window, there are 'Cancel' and 'Continue' buttons.

[Source: Firmbee - Timetracker](#)

Therefore, digital means of creating a Burndown Chart are much more common. Many tools for logging work on tasks distributed among Team members come with an option to automatically create a Burndown Chart. Then, all a Developer has to do is mark the start and end of work on a particular product feature, and their contribution is reflected in the Burndown Chart.

With the right tools, it is also possible to freely scale the chart. This gives an insight into the combustion not only on the level of a given Sprint but also on the scale of a quarter or the whole project. **An important factor to consider when choosing a tool for creating a Burndown Chart is its accessibility to all Scrum Team members.** The visibility of the Burndown Chart to the entire Development Team is a key motivational factor. Equally important is a daily look at the line showing the work remaining to be done. Talking about burn-in during the Daily Scrum, gets Developers thinking about the ways they work and the current state of the Product.

Who is responsible for the Burndown Chart?

The question of ownership of the Burndown Chart is somewhat controversial. On one hand, it should belong to the Scrum Master, because it is a tool to make sure that the Team is working efficiently and according to plan. On the other, it should remain in the hands of the Product Owner, because it reflects the progress towards a Product Goal that's communicated to the Customer. What's more, a third party to claim its ownership is the Development Team as the chart functions as its internal tool.

The Burndown Chart is an essential metric to evaluate the effectiveness of the Development Team and becomes adopted by all Scrum Team members. That's why transparency and accessibility are crucial. However, its very purpose is to serve the Team. It is supposed to strengthen its self-organization, improve motivation, and give a real picture of the status of work on the tasks assigned to it. Therefore, in theory, each member of the Development Team can update the Burndown Chart.

In practice, however, the task of updating the Burndown Chart usually falls to the Scrum Master. This happens especially at the beginning of his work with a new Development Team when the Team Velocity is still variable and difficult to estimate. Nevertheless, it is recommended to delegate this task to one of the Developers. After all, the chart is meant to be an honest and internal measurement of the progress of the work as judged by the Developers themselves.

Real and ideal Burndown Chart

To interpret a Burndown Chart, the key factor is not only the regular plotting of the real "combustion", i.e. the execution of tasks by the Development Team. Equally important for the picture is its comparison with the ideal combustion line drop (guideline).

By comparing the ideal burn line with the real-world reduction in work marked on the Burndown Chart, two very important parameters can be assessed. First, to see whether the work continues at the current pace, the Development Team will meet the Sprint Goal or Product Goal on time. Second, to get an idea of when the work will be completed while maintaining the current pace. In other words, the Burndown Chart shows the actual pace of the tasks, and the ideal line shows at what pace the Team should work to complete the tasks.

The Burndown Chart also allows you to determine a value called Development Team Velocity in the long run (it is a value determined by the amount of work done during one Sprint). Thanks to the fact that the Burndown Chart illustrates the comparison of an ideal burn line with a real decrease in the number of tasks, it allows you to estimate the pace of work and thus anticipate the risk of project delays.

Selecting the unit of measurement

Team Velocity is usually measured in units called Story Points. It defines the number of User Stories that have been realized. These can require very different amounts of work. This is why many Scrum Teams use a time-based measure. Depending on the scale, these are days or man-hours. Each Developer estimates and then logs the amount of time spent on their tasks.

Another option is to adopt tasks as a unit. These are slightly larger units, which in turn are assigned a value expressed in Story Points, or in days or man-hours. It is a unit that allows the client to present the progress of work on the product in a clearer way. Regardless of the unit of measurement, it is worth remembering the principle of calculating the speed of the Development Team. In a given day or Sprint, only tasks that were actually completed, count. It means that tasks started will be counted towards the next day or Sprint even if only final testing is missing.

Pros and cons of the Burndown Chart

The Burndown Chart has many benefits. It is one of the main metrics tools in Scrum for several reasons. It's easy to create, scale and read. However, it also has drawbacks that make it not a universal tool. Let's cover the topic of disadvantages and advantages of the Burndown Chart. However, most of them are not hidden in the simple graph itself. Rather, they are related to the ways of using the Burndown Chart to motivate the Development Team, as they describe the results of their work and strengthen self-organization.

1. Advantages of the Burndown Chart

The Burndown Chart allows you to visualize the progress of your project. Its readability and simplicity make it so popular. That's why it's a good idea for the Burndown Chart to be not only a constantly updated metric hidden in a digital project management tool. If at all possible, it's worth making it a reference point for the Development Team visible in the physical workplace. Whether in the form of an on-screen visualization or a hand-drawn sketch.

● It motivates the Development Team

The transparency of the Burndown Chart can make it a tool to motivate the Development Team to work efficiently. **Reaching the "zero" point in each Sprint can become an ambitious goal of the Team**, for which rewards are given – according to the principles of business gamification. The visibility of an up-to-date and interestingly maintained Burndown Chart can also enhance a spirit of cooperation and self-organization. After all, the metric is a measure of teamwork. It doesn't show exactly who did – or didn't – complete the planned tasks, only the results achieved.

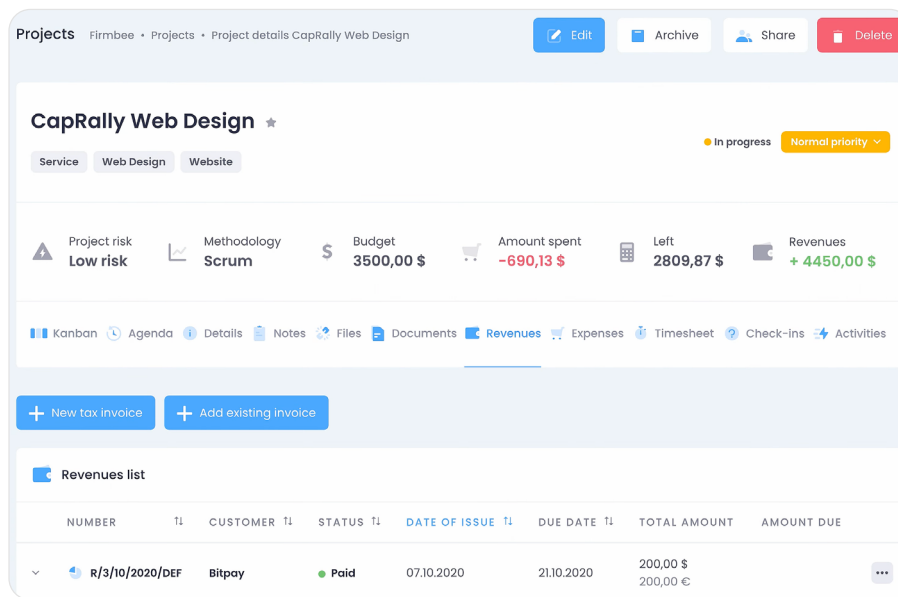
● It measures the actual work performed

Developers decide how many tasks they will perform in a given Sprint. The more experienced the Team is, the more accurately they should predict their actions. And the Burndown Chart reflects the real progress of the Sprint. Thus, **the advantage of the Burndown Chart is not so much to measure the goal amount of work done, but the ratio of planned-to-completed tasks**. Thus, Developers gradually learn how to plan them and can estimate their capabilities more and more accurately and eliminate repetitive errors.

● It couples with other tools

One of the significant advantages of the Burndown Chart concerns its versatility in combining with other tools. The following tools can apply to: analyzing the work of the Development Team, visualizing the progress of work on the Product, estimating the project budget. For example, in the latter case, the use of the Project Scale Burndown Chart allows for a comparison of the planned and actual budget for the entire project.

The Scrum team can track receipts and expenditures in real time in the project management system. In this way, each member will be able to control the budget for a given project.



[Source: Firmbee - Project budget management](#)

2. Disadvantages of the Burndown Chart

Despite all the advantages of the Burndown Chart outlined above, it can become a source of confusion for the Development Team. However, what we frequently call the “flaws” of the Burndown Chart are not due to imperfections of the tool itself. The problems outlined below concern the way of implementing the Burndown Chart rather than its design. Below are the flaws that can interfere with depicting the progress of the Development Team in this way.

🟡 “The Human Factor”

Charts cannot be an absolute measure of a team’s progress. They are just tools to apply in different, more or less skillful ways. We can consider it as a disadvantage (or advantage) not only of the Burndown Chart but also of other measures of team performance.

To create a Burndown Chart, you need other people to enter data. In other words, the Developers put down task completion time on the chart. They might have lengthened or shortened it a bit – either through inattention or wanting to make things better for the team. Developers also sometimes forget to log their time or leave the timer on. This causes the working time to extend to several hours and after discovering the mistake, it is difficult to reconstruct its real course.

🟡 Changes to the Sprint Backlog

Sprint Backlog should not be modified after the start of a Sprint. However, in practice, such changes occur quite often. They result from changing requirements of Stakeholders or unforeseen problems that Developers encounter. This causes the Burndown Chart to be scaled. This is because the time taken to

complete the tasks remains the same. However, the scale of tasks remaining increases. This may give a misleading impression that the Development Team has incorrectly planned the work to do in a given Sprint or that it works too slowly.

Changes to the Sprint Backlog can also result from tasks that were scheduled for completion too quickly. In such a situation, the Development Team usually decides to increase the number of tasks. This in turn may result in not completing them on time. Also, conflicts may arise from the overlap of remaining tasks from the previous Sprint with new tasks scheduled to be completed by Stakeholders and Product Owners.

🟡 Changes to the Product Backlog

Big changes in the Product Backlog can disrupt the shape of the Burndown Chart. And thus strongly falsify the picture of work progress and Team effectiveness. This happens when new User Stories appear. And those which are close to the implementation phase are often broken into smaller parts. It also happens that the Client resigns from some Product functionalities.

Therefore, when interpreting the Burndown Chart, one must be guided by knowledge and experience in evaluating the Team's performance and also take into account the variability of the Backlog. If the Burndown Chart is not the only metric used to evaluate performance, the other charts will allow you to see a more complete picture of the progress of the work.

Scrumban and Kanban boards in Scrum

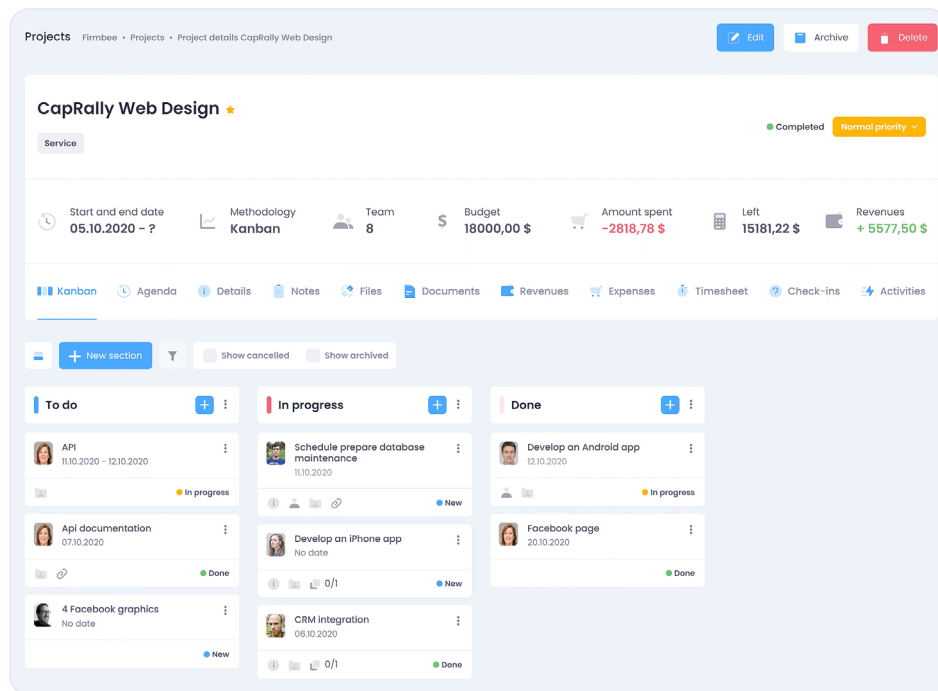
Scrum and Kanban are teamwork methods that share many similarities. However, there are also differences that we would like to discuss today. Kanban boards are also often adopted by Scrum Teams. This is because they are very practical in visualizing teamwork and its progress. By combining the best of both methodologies, there appeared a technique called Scrumban. It is popular in projects that combine Product development with service delivery, where long Sprints and relatively formalized Scrum meetings are not always suitable.

Kanban is a method pioneered in Japan. It originated in the 1950s and was primarily a tool for managing continuous production in such a way as to not create inventories and surpluses, but to process resources on an ongoing basis. In the early 21st century Kanban was adapted to the needs of software development by David J. Anderson.

Kanban vs Scrum

The overall way of working in Kanban differs from Scrum primarily by bringing about a less formal approach. In Kanban, there are not so detailed guidelines on, for example, working in Sprints, roles of Product Owner, Scrum Master and Development Team. This is possible because Kanban focuses on the continuity of tasks such as providing a specific type of service, which are more repeatable and does not require such complex planning. However, the purpose and ways of working are similar. **The goal of Kanban is to deliver the highest quality product to the customer on time.** The principles concerning the ways of working common to both methods can be formulated as follows:

- ⬢ **The work should be smooth and without any downtime** – in Scrum, this is achieved by the continuous succession of Sprints, while in Kanban the work is continuous due to the smooth flow of tasks. They form a queue, from which the Developers choose (pull) a few tasks to complete.
- ⬢ **The team should focus only on selected tasks** – using Kanban terminology, the team should “reduce work in progress”. In Scrum, the equivalent of this is User Stories chosen from the Product Backlog to put into the Sprint Backlog
- ⬢ **Progress of tasks should be visible for all people involved** – in Kanban they are visualized by boards, which are also often featured in Scrum Teams.



Source: Firmbee - Kanban boards

Kanban Boards in Scrum

A Kanban board is a widely used tool for visualizing teamwork. It is a table with several columns. In each of them, there are tasks with a certain status. Categorization of tasks is based on a simple rule: a card with a description of the task – or its virtual equivalent – is placed in one of the columns. The minimum version of Kanban boards contains three columns: To do, In progress and Completed – to the last column go the tasks that meet the Definition of Done.

Commonly, there are more columns. If there are more tasks to complete, there is usually an additional column titled “selected for completion” between the “to be completed” and “in progress” columns. While the “to-do” column serves as the Product Backlog the “selected for completion” column serves as the Sprint Backlog.

[Source: Firmbee - creating new section in Kanban board](#)

The second common addition is an “under review” column or “for approval”. It is usually inserted between the columns containing the “in progress” tasks and the “completed” ones. It contains tasks completed by the Development Team that is waiting for approval from the Product Owner. The Product Owner’s task is to check their compliance with the Acceptance Criteria and get their final approval from the Customer. In this situation, only the finally accepted tasks are moved to the last column.

Scrumban

Due to the huge popularity of Scrum and Kanban, their hybrid appeared, combining the best of both ways of working. **Scrumban works best in organizations that connect the creation of Products with the provision of services, often involving the implementation of the Product at the Customer.** Because of the reduction in meetings and communication, the Team can be larger.

Scrumban places less emphasis on metrics commonly used in Scrum, such as the Burndown Chart. However, it uses the Scrum pillars of the need for continuous improvement of the work process and adapting them to the customer’s conditions and needs. When working in Scrumban, however, the work is not divided into Sprints. Scrum meetings are held every 3, 6, or 12 months.

Scheduling of work follows the “On-Demand” principle, i.e. as it occurs. User Stories are placed directly in the first column of the Kanban board containing “to-do” tasks. Thus, it serves as the Sprint Backlog. As in the Sprint Backlog, the most urgent tasks are placed at the top of the to-do list. However, for more complex projects, the Project Manager can maintain a separate to-do list corresponding to the Product Backlog, from which he selects which tasks to place in the first column.

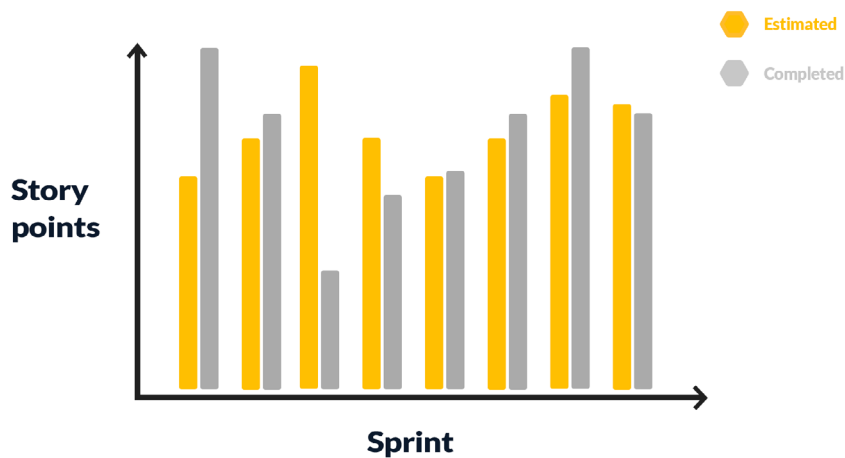
When moving tasks from the first to the second column, the “Pull” rule applies. It means that tasks are not delegated to a particular Developer. Each person chooses a task from the queue and executes it independently. The number of tasks placed in the middle column, “to complete” is usually limited depending on the size of the team, so that, if possible, everyone deals with only one task at a time.

Velocity in Scrum

Velocity in Scrum helps you to determine the rate at which the Scrum Team completes tasks. We can define it as the average number of Story Points completed in one Sprint. Velocity can also estimate the duration of a project based on work progress already completed. However, this only makes sense for a mature team that works at an even and steady pace.

Velocity is an optional but popular method to measure the pace of a Scrum Team. This is because **an accurately estimated Velocity enables predicting, to a reasonable extent, the time needed to complete a project.** However, it is a measure that can only be applied to a given Development Team, which will perform tasks that it has “valued” itself using a familiar unit, such as Story Points, for example.

The Velocity of the Development Team is most often presented in the form of a Velocity Chart. On the X-axis are marked consecutive Sprints. On the Y-axis, on the other hand, we will find the number of Story Points or other corresponding units that were completed in a given Sprint. With the Velocity Chart, the Scrum Team gains a clear view of changes in the pace of its work. If the line marked on the chart is rising, it means that the Team is optimizing its efficiency or reducing the value of Story Points. Both the Scrum Master and Product Owner should therefore carefully follow the line showing the Team's Velocity.



Actual and planned Velocity

The actual Velocity of the Development Team describes the pace of work in the completed Sprint and is calculated at the end of each Sprint. It takes the value of the sum of Story Points for all completed User Stories. The actual Velocity of the Development Team allows you to plan and estimate with some probability the pace of future tasks.

The planned Velocity, on the other hand, is estimated based on an average value of the actual Velocity. It requires the assumption of no change in the Development Team. It is an important internal tool for the Development Team, which, based on it, can assess whether cooperation in the Team is going well and whether the pace of work is being maintained. Planned Velocity also enables the Product Owner to

forecast the execution time of well-defined User Stories scheduled for execution in subsequent Sprints. This enables more efficient nurturing of the Product Backlog. However, the practice of applying planned Velocity to estimate project durations is not so simple.

Difficulties and risks associated with Velocity in Scrum

Velocity in Scrum is often given too much importance without considering the following factors:

- **estimating larger wholes or the entire project** – while the Development Team can accurately estimate the number of Story Points to be assigned to a specific task, it is very difficult or impossible to describe larger wholes for future implementation in these units,
- **changes in the project** – any change in the project potentially means a change in the number of Story Points needed to achieve the Product Goal. It may also be that tasks already completed will need to be modified or even not used in the final version of the Product,
- **unforeseen events** – predicting the pace of future projects based on those already completed, that is, translating actual Velocity into Planned Velocity, can result in accurate estimates. However, each project has its peculiarities and an accurate prediction based on history is usually impossible.

Daily Scrum

The Daily Scrum lasts no more than fifteen minutes and is always held in the same place and at the same time to reduce unnecessary complexity. It is attended by all Developers working together on the Product and, optionally, the Scrum Master. The main purpose of this Scrum Event is to plan the tasks they will focus on for the day. Daily Scrum is the shortest and most frequent of the Scrum Events. The task of Developers participating in Daily Scrum is to quickly set work goals for the next 24 hours. This way, each of them knows what the others are working on and how they are working towards a common Sprint Goal.

The Daily Scrum formula

There is no one right Daily Scrum formula. Each Development Team develops a meeting format that works for it. However, there is a general framework to make it easier to conduct. A well-conducted Daily Scrum should allow each participant to answer two questions:

- What is the most important task I will perform today?
- What are the obstacles to accomplishing this task?



However, asking them directly is not a mandatory formula. These are sample questions that define the axis of the meeting. **Daily Scrum is intended to improve communication in the Development Team, prioritize tasks and reduce the risk of bottlenecks.**

The Daily Scrum is an event equivalent to the Daily Standup in other Agile methods. And it often runs very similarly to it – although the official Scrum Guide does not require Developers to stand during this short Event. Very often its participants simply stand while talking in an informal group. While it may seem that 15 minutes a day is a lot for discussing daily tasks, practice shows that such a meeting is best for the effectiveness of the Development Team. With frequent and regular updates on goals and commitments, all Developers focus on priority tasks and prioritize smooth team progress over individual results.

Problems with Daily Scrum and the 5W method

One of the problems with Daily Scrum is that Developers drag out the meeting time. If this is the case, it's a good idea to introduce a policy of writing down on a board – either physical or virtual – problematic issues that are not central to the Daily Scrum but are important to the Team. In this way, it will be possible to return to the problems that were left to be discussed during the informal discussions during the day and also, if needed, during the Sprint Retrospective.

Another problem that often arises during Daily Scrums is turning them into meetings to summarize the previous day's work. Developers then focus on discussing the results already achieved. This is not a good practice. Admittedly, the current orientation of Developers on the status of the work leading to the Sprint Goal is very important. However, devoting the Daily Scrum to already completed tasks does not promote efficiency.

Supporting questions

If the Team is not benefiting from the Daily Scrum, the Scrum Master can help Developers identify problems by observing the meeting for answers to the following questions:

PROBLEM AREA	QUESTION
Understanding the Sprint Goal	Is the Sprint Goal clear for Developers?
Participation in the meeting	Does each Developer have the opportunity to speak freely?
Mutual respect	Do Developers listen to each other?
Teamwork	Do Developers help each other?
Commitment to problem-solving	Do Developers offer different ways to solve problems that arise?

After the initial identification of the problem, an effective technique for determining the cause of the problem can be the 5 Why method also called 5 Whys or 5W by Sakichi Toyoda. It involves asking several "Why?" questions in a row. This makes it possible to diagnose the deeper cause of the problem, and thus solve it more easily. For example, let's take the last item in the table: the problem arises in the area of commitment to problem-solving by the Development Team. The five questions might look as follows:

1 x WHY?

Q: Why don't Developers offer different ways to solve problems that arise?

A: Because Developer Harry is always the first to propose one solution.

2 x WHY?

Q: Why is Developer Harry always the first to propose one solution?

A: Because no one else is speaking.

3 x WHY?

Q: Why doesn't anyone else speak up?

A: Because other Developers have no desire to look for better solutions.

4 x WHY?

Q: Why don't other Developers feel like looking for better solutions?

A: Because finding solutions requires focus and it is easier to consider Harry's solution good enough.

5 x WHY?

Q: Why did they consider Harry's solution good enough?

A: Since they are not rewarded for proposing alternatives, they discussed their plans for today at the beginning of the meeting and are thinking about getting started.

In this case, the problem of lack of commitment to solving problems can be solved by changing the order of the Daily Scrum and starting with this issue or coming up with a system for rewarding the best solution, for example, introducing a symbolic reward for the author of the largest number of solutions accepted by the Team in a given Sprint.

Sprint Planning

Sprint Planning is one of the Scrum Events that centers around the User Stories placed at the very top of the Product Backlog. In other words, on those that are most detailed. The role of the Product Owner during Sprint Planning is to present the Goal and the tasks comprising it to the meeting participants. The role of moderator of Sprint Planning is usually performed by the Scrum Master. The flow of a textbook meeting can be summarized by answering three questions:

1. What is the new Sprint Goal?
2. What is on the agenda?
3. How will the team do it?

Let's take a closer look at what the answers to the above questions should include.

1. What is the new Sprint Goal?

The role of the Product Owner during Sprint Planning is to present the Goal and the tasks comprising it to the meeting participants. **The Product Owner starts the meeting by formulating the Sprint Goal and justifying why it is valuable from the customer's point of view.** Then opens a discussion in which not only Scrum Team members, but also Stakeholders can voice their opinion. To sum up, the Product Owner gives the final wording of the Sprint Goal that the entire Scrum Team will strive to achieve and makes sure that the Goal is understood by all Stakeholders.

2. What is on the agenda?

The second part of Sprint Planning focuses on selecting the User Stories to implement in the new Sprint and discussing how to make them more specific. **One of the most difficult tasks during Sprint Planning is to accurately estimate the number and labor intensity of the tasks selected for execution.** The more experienced the Scrum Team, the more accurate it can estimate how much work can be done in a single Sprint. This is because the Team is applying effective estimation techniques. Many Scrum Teams know and utilize methods to speed up the maturation of the Team as well as make the process easier and more standardized. These techniques include primarily Planning Poker and Team Estimation Games.

3. How will the team do it?

The third and most technical part of Sprint Planning focuses on answering the question "How will the team do it?". **In it, the Development Team proposes ways to accomplish the tasks selected for implementation in the second part of the meeting.** No one but the Developers themselves should dictate how the tasks should be executed from the technical side.

Planning should take into account not only the execution technology but also the workflow between Developers. This will avoid stagnant work [bottlenecks], which can cause delays in the execution of tasks. As in the case of how to execute tasks, the distribution of tasks among individual Developers is also decided by them alone without external interference. Typically, the Development Team here focuses on dividing User Stories into smaller tasks. The optimal length of task execution is one working day.

The result of Sprint Planning is an unambiguous Sprint Goal, as well as detailed User Stories selected for execution from the Product Backlog. All these elements make up the Sprint Backlog.

Sprint Review

Sprint Review is a Scrum Event that summarizes the work on the Product that was completed during the current Sprint. It takes place on the last day of the Sprint and is open to Stakeholders. Its purpose is to evaluate the Increment, i.e. to present the latest version of the Product. An important part of the Sprint Review is also the discussion of the improvements and updates made and also to make the necessary changes to the Product Backlog, so that all Stakeholders can see the current status of the Product.

Sprint Review is dedicated to the Product and its purpose is to inspect the Increment, i.e. the results of the work done in the Sprint that just ended. The event lasts a maximum of four hours. All members of the Scrum Team, as well as Stakeholders, i.e. all people interested in the progress of the Product attend it.

The role of Stakeholders during Sprint Review

During the Sprint Review, the Scrum Team presents the Increment to the Stakeholders. In doing so, it summarizes the completed tasks and answers specific questions:

1. Who performed the task?
2. What specifically was done?
3. For what purpose was it done?

You will find the answer to all these questions in Firmbee. You can track the progress of your work in real time, monitor the activity history of individual team members and assign them specific tasks. You also have a constant overview of the revenues and expenses related to a project.



However, the most important part of the Sprint Review is the discussion with the Stakeholders regarding the Increment, as well as the direction in which the Product is being developed. The Stakeholders provide feedback to the Scrum Team members. This allows for adaptation, i.e. adjusting the Scrum Team's way of working to the needs and vision of the Customer. This is done to maximize the business value of the Product. The feedback provided at each Sprint Review is particularly important when creating innovative products that need to be adapted on an ongoing basis to the activities of the competition and the needs of the market.

Releasing Increments

We shouldn't consider Sprint Review as the only time the Scrum Team releases an Increment to the Customer. If some Product functionality meets the Definition of Done beforehand, the Product Owner may decide to release it immediately. It is also possible that an item of the Product Backlog that the Scrum Team worked on in a given Sprint has not been completed and does not meet the Definition of Done. It cannot then get released or even presented during Sprint Review.

Working on the Product Backlog during Sprint Review

Updating the Product Backlog is as much a part of Sprint Review as presenting the work results to Stakeholders. Usually, the Backlog update is dedicated to the last part of the meeting, so Stakeholders do not have to attend.

The Product Owner updates the Product Backlog based on feedback from Stakeholders and lessons learned by the Development Team. This is especially crucial if the feedback obtained has an impact on the shape and purpose of the next Sprint. Updating the Backlog is then an essential step to preparing the next Sprint Planning.

Sprint Retrospective

Sprint Retrospective is a Sprint wrap-up event that only Scrum Team members can attend. This allows it to be fully dedicated to the internal affairs of the team. This is because the Sprint Retrospective is primarily used to reflect on current working methods, as well as to discuss suggestions for improving them. Sprint Retrospective is the meeting that ends each Sprint and it is one of the Scrum Events. According to the official

Scrum Guide, a Sprint Retrospective takes a maximum of three hours for a monthly Sprint or correspondingly shorter if the Scrum Team works in shorter cycles.

Goals and topics of Sprint Retrospective

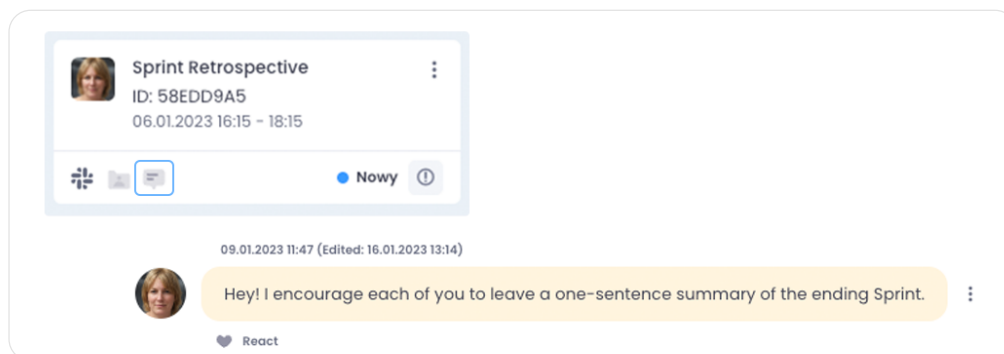
All members of the Scrum Team take part in the Sprint Retrospective. **The purpose of the meeting is to discuss problems related to the Scrum Team's work and how it handles them.** However, these are not problems related to the Product being developed by the Scrum Team, but issues related to the nature and course of cooperation between Scrum Team members. Because the issues raised are often sensitive and touchy, the Sprint Retrospective is a closed event. We can formulate its goals in the following way:

- to summarize the current ways of cooperation,
- to identify those problems and imperfections that require improvement,
- to suggest solutions and modifications.

The goals of Sprint Retrospective are closely related to the pillars of empiricism on which Scrum is supported. The first two points are related to inspection. While the last one is related to adaptation. The result of the answers to the above meetings is not only a clear picture of the Scrum Team's principles of cooperation available to all its members. The Team also makes commitments to improve cooperation and team behavior, which will be implemented in the next Sprint.

How to conduct an effective Sprint Retrospective?

Because the Sprint Retrospective is a difficult meeting, the role of the Scrum Master who moderates the discussion is crucial. Ideally, he should suggest to the Scrum Team members to speak next. For example, he can ask everyone to give a one-sentence summary of the ending Sprint.



[Source: Firmbee - Comments](#)

1. Problems to be discussed

Since talking about problems in the team can stir up a lot of emotion, a common solution is to write down the issues to be discussed on separate pieces of paper. This makes it easier to express your opinion. It's also easier to spot larger problem areas and issues that more people have concerns about. If there are too many issues that Scrum Team puts forth, you can start by discussing the major ones or select collectively which issues are the most important in the opinion of the Scrum Team. You can postpone problems for which there was not enough time during the Sprint Retrospective to the next retrospective. Of course, only in case they still occur.

2. Discussion and commitment

The most important parts of the Sprint Retrospective, however, are discussion and making commitments.

The discussion should focus on the causes of the problems, the moments when they occur, and their impact on the functioning of the Scrum Team. It is worth considering whether their occurrence can be avoided and with whom to discuss their solution.

Making commitments is just as important as diagnosing the problems because just knowing they exist and the causes do not translate into solving them. **The result of a Sprint Retrospective is usually several commitments.** If the problem affects the whole team, often one of the team members is committed to paying special attention to a particular problem in the next Sprint. And to propose its solution, or even to solve the problem itself. If, on the other hand, the problem concerns the action of a specific person, he commits to change his behavior as early as the next Sprint.

Common mistakes during a Sprint Retrospective

Mistakes during a Sprint Retrospective are unfortunately very common. This is because it is one of the most difficult meetings to successfully execute as it requires a lot of maturity from the team. That's why it's worth taking a look at the problems that occur most often in other teams so that you can more easily spot their symptoms when conducting Sprint Retrospective in your Scrum Team.

1. Insufficient transparency

According to the Scrum Guide, each member of the Scrum Team is obliged to be honest and bold in expressing concerns and to voice their opinion during the Sprint Retrospective. However, in practice, the commitment to transparency is very demanding. Because of this, Scrum Team members often try to circumvent it. **One problem that is difficult to spot and solve is avoiding discussion of observed shortcomings in the Scrum Team's work.** This can lead to much more serious problems in the long run. The Scrum Master's task, therefore, is to keep a close eye on the situation in the team and encourage all team members to be proactive from the very beginning of the Sprint Retrospective.

2. Focus on one-time problems or successes

Another problem that can arise during Sprint Retrospective is paying insufficient attention to cyclical and repetitive team behaviors, and their impact on team effectiveness. It's always good to congratulate Scrum Team members if they have achieved exceptional success. However, Sprint Review should not be dedicated to celebrating it. The same is true of failures. If something failed due to fortuitous reasons or an already diagnosed error, it is not worth over-analyzing the event during Sprint Review.

Sometimes, however, the team devotes a large part of the Sprint Retrospective to such events. Keep in mind though, that the purpose of Sprint Retrospective is to look for ways to improve the team's daily work. Therefore, the meeting should not revolve around one-time successes or problems that are highly likely not to happen again.

3. Over-representation of Product Owner

In many organizations, the position of Product Owner is equated with that of Product Manager. Product Owner is then often considered the supervisor of the Scrum Team. For this reason, it happens that the Development Team does not want to talk about teamwork problems in his presence.

That is why it is so important to build mutual trust between the Development Team and the Product Owner. Unfortunately, the process of building trust is difficult and lengthy. That's why sometimes it's a good idea for the Product Owner to give up participation in all or part of the Sprint Retrospective to leave space for the rest of the team to discuss freely.

4. Self-management problems

Self-management means that Scrum Team members make their own decisions about who among them will perform certain tasks, when and how. During Sprint Retrospective, the team discusses people, their interactions as well as team practices. It then decides what problems need to solve in the upcoming Sprint, how to do it together with who will bear responsibility for taking action.

If more serious problems arise in a self-managing team, there may be a temptation in the Scrum Team to abdicate responsibility. Occasionally, team members don't want to take part in the discussion and try to push the management responsibility onto someone else. To prevent this, it is extremely important to discuss even small problems regularly to prevent their accumulation.

5. Too many commitments

An active Scrum Team operating following the three pillars of empiricism: transparency, inspection and adaptation, may encounter the problem of making too many commitments at once. If the commitments made by the Scrum Team during a Sprint Retrospective are too many, there is a considerable risk that: none of the commitments will be implemented properly, some commitments will not be implemented at all or the changes made will not be permanent. Therefore, a good practice is to undertake no more than four improvements in each Sprint. This allows for a gradual but effective improvement of the team's performance.

5. Where to get knowledge and experience in Scrum?

Have you already read our e-book? Do you feel like you need more? If you want to deepen your Scrum knowledge and maybe even get certified, the following text will provide you with proven sources, as well as information on official Scrum certification.

The primary source of Scrum knowledge is, of course, *The Official Guide to Scrum* by Ken Schwaber and Jeff Sutherland. It is available for free at this [link](#). Its latest and finest version from 2020 has been translated into more than 30 languages. It includes a description of all the basic principles of Scrum and an explanation of the necessary concepts in a concise, minimalist form.

1. Scrum knowledge

The list of guides and tutorials on working in Scrum grows every year. Among them, you can find both books intended for beginners and those discussing detailed problems encountered by people working as Scrum Master or Product Owner. We decided to mention here three general and most frequently recommended books on Scrum:

- ◆ **Scrum: The Art of Doing Twice the Work in Half the Time** - This book, full of examples, by Jeff Sutherland, centers around the topic of changing the world, and in particular our everyday environment - the workplace. He emphasizes that Scrum works not only in business but also in education or in solving social and personal problems. In other words, the type of project is often not as important as the way we think about its implementation. The ability to adapt to changing conditions and learn from mistakes is actually the key.
- ◆ **A Scrum Book. The Spirit of the Game** - One of the co-authors of this book is also Sutherland. Its main purpose is to describe the patterns of Scrum. By knowing them, you can both spot patterns more easily during the implementation of successive projects in Scrum and get better at making improvements to projects.
- ◆ **Scrum Mastery. From Good to Great Servant Leadership** - Geoff Watts' book is a practical guide describing how to work on improving the skills necessary to be a Scrum Master. He also places great emphasis on describing the difficult realities of working in Scrum in organizations that have previously used traditional management methods.

An invaluable source of knowledge, but also an opportunity to discuss the practical applications of Scrum is the forum available at [Scrum.org](#). It is shared by people who are just learning and looking for answers to practical questions that they can't answer from the Scrum Guide. And also experienced Scrum practitioners willing to share their knowledge about working with a Scrum Team, implementing Scrum in an organization, or details about the work of a Product Owner or Scrum Master.

Those interested in applying Scrum principles to larger organizations should take a look at the version of the [Scrum@Scale Guide](#) also prepared by Ken Schwaber and Jeff Sutherland. Like the Official Guide to Scrum, this manual has also been translated into many languages. In it, you can find not only the principles of scaling Scrum but also a detailed description of the benefits of using Scrum in larger organizations.

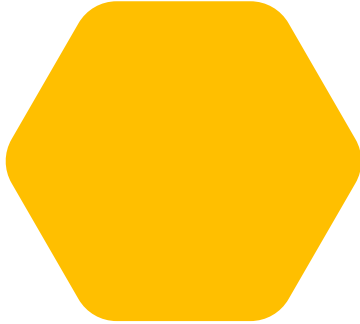
If you're interested in how Scrum compares to other Agile practices, you can reach out to the [State of Agile Report](#). This is one of the largest reports on Agile, which has been produced annually since 2006. Among other things, you'll learn from it what Agile techniques and practices are being used in companies and also what the spread of working techniques like Scrum from the IT industry to the business world as a whole looks like and how quickly it proceeds.

2. Scrum training and certifications

Gaining knowledge about Scrum becomes easier with online courses. If your organization is willing to send you for training, there are also onsite classes. The best source of knowledge is the certified Scrum training courses available [here](#). They allow you to gain in-depth knowledge about practicing Scrum principles in your organization.

You can take a training course on how to implement Scrum, or take a course leading to a professional Scrum Master or Product Owner certification. Scrum.org also provides training on combining Scrum principles with the use of Kanban boards and also on scaling Scrum or its combination with user experience (UX) design.

After completing the training, you can obtain the valued certifications that are so important on the resumes of those applying for senior positions. You can get the following certifications as proof of professional, practical skills: Scrum Master, Product Owner, Scrum Developer, as well as Scrum@Scale Implementation Professional. Considering the effectiveness of working in Scrum, as well as the translation of practical skills for its implementation into earnings, it is worth obtaining a professional certification.



Get to know us better

