# Origin

In rugby, a scrummage (or *scrum* for short) was the method used to restart play in a match after a foul. Visually, it's eight players from each team packed together with heads down, all trying to take possession of the ball. So, not exactly the poster child for project management, but with a little imagination, it makes sense.

On a project team, the goal is to get the project done. Historically, **in traditional methods, we plan and design the whole project at the beginning and stick to that plan with no variation**. However, modern project work is completely unpredictable. It's impossible to know at the beginning exactly how a project will unfold and how to best meet its unique challenges.

The Agile Project Management comes out an idea to adapt in real time to the changing circumstances, and this is where the rugby team comes in. In rugby, the object is to move the ball down the field, one possession at a time, so why couldn't projects do the same thing? Why not change the focus from just winning the whole game to winning each and every milestone and deliverable?

The scrum reflects this approach – **breaking the deliverables and milestones into smaller pieces and gets the whole team together to focus on just that one goal until it's done**.

# Agile Overview

## Manifesto

* **Individuals and interactions** over processes and tools
* **Working software** over comprehensive documentation
* **Customer collaboration** over contract negotiation
* **Responding to change** over following a plan

## Principles

1. Our highest priority is to **satisfy the customer through early and continuous delivery** of valuable software.

*We not just show up at the beginning to describe what they want, then show up at the end and tell us how we missed the mark. We need direct, ongoing interaction.*

1. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
2. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
3. Business people and developers must work together daily throughout the project.
4. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
5. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
6. **Working software** is the primary measure of progress.

*We no longer want to measure success using milestones and phases. We want working software to tell everyone how we're doing, and we want to hear feedback the whole time.*

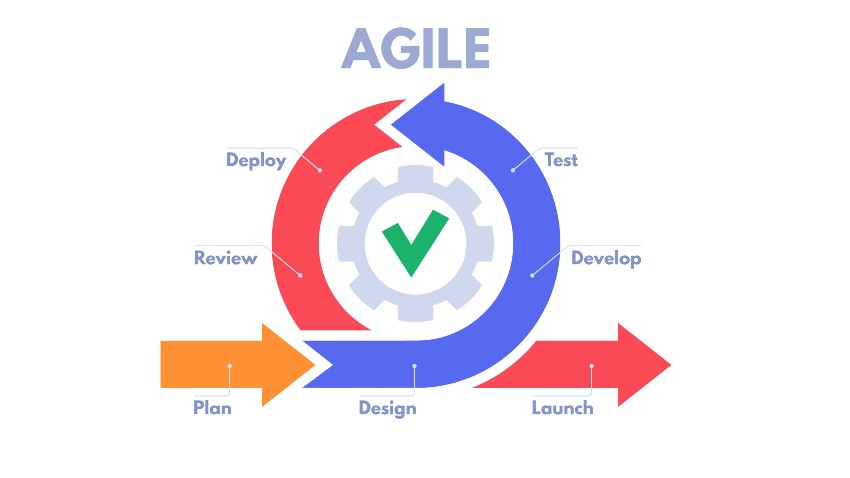
1. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
2. Continuous attention to technical excellence and good design enhances agility.
3. **Simplicity** – the art of maximizing the amount of work not done – is essential.
4. The best architectures, requirements, and designs emerge from **self-organizing teams**.

*Allow teams to self-organize. They'll do a much better job doing the design and tests from the ground level than any upfront plan could do.*

1. At regular intervals, the team reflects on how to become more effective, then tunes and **adjusts its behavior accordingly**.

## Workflow

A spiral model:



## Agile Frameworks

Scrum, Kanban, Scrumban, EXtreme Programming (XP), Crystal, Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), Adaptive Software Development (ASD)

# Scrum Overview

Scrum is the **most common Agile framework for effective team collaboration on complex software projects**. It’s founded on empiricism (*chủ nghĩa thực nghiệm*) and lean thinking (*tư duy tinh gọn*):

* Empiricism asserts that **knowledge comes from experience** and **making decisions based on what is observed**.
* Lean thinking **reduces waste and focuses on the essentials**.

Scrum employs an **iterative, incremental approach** to optimize predictability and to control risk. It engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

## Methodology

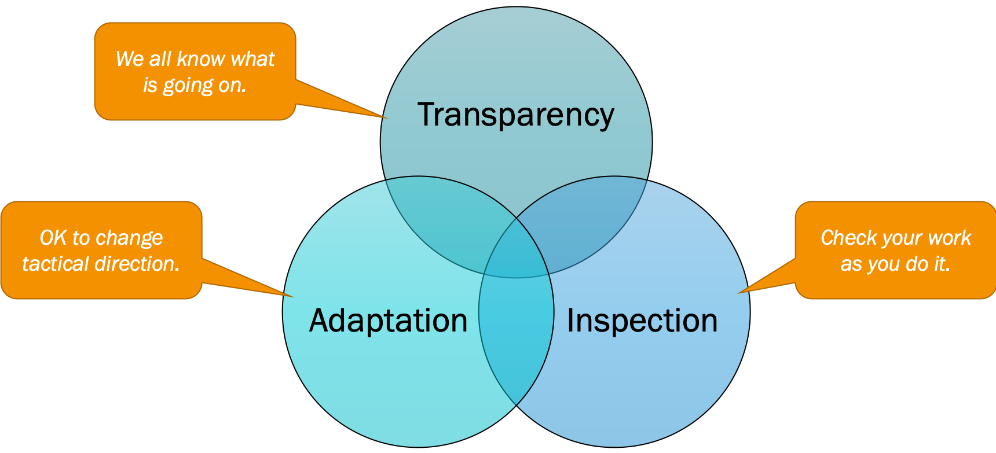
Scrum wants you to fail. In fact, it's known for the slogan, "**Fail fast**." Although it sounds weird, there's a very good reason for this.

Traditionally, project managers and developers would work for months or years before seeing the results. Most of the time, around 80%, the software and projects failed.

The trick is in focusing on the second word, "fast". **Failure is okay, as long as you're learning from it, but if you have to wait too long, you're not going to learn nearly as much from i**t. Scrum takes the agile manifesto and its key principles and boils them down to a very simple framework that

The Scrum is designed to encourages small scale focus and fast learning cycles. That's what fail fast really means: "**learn fast**."

## Scrum Pillars



### Transparency (minh bạch)

The **work must be visible to those performing it and those receiving it**. This means presenting the facts as it is. All people involved should trust each other. They have the courage to keep each other abreast of good news and bad news. Everyone strives and collectively collaborates for the common objective, and no one has any hidden agenda.

Transparency enables *inspection*. Inspection without transparency is misleading and wasteful.

### Inspection (kiểm tra)

The progress toward agreed goals must be **inspected frequently and diligently to detect potentially undesirable problems**.

For example, the team transparently shows the product at the end of each Sprint to the customer in order to gather valuable feedback. If the customer changes the requirements during inspection, the team does not complain but rather adapts by using this as a chance to collaborate with the customer to clarify the requirements.

Inspection enables *adaptation*. Inspection without adaptation is considered pointless. Scrum events are designed to provoke change.

### Adaptation (thích ứng)

Adaptation is about continuous improvement, the ability to adapt based on the results of the inspection. If any aspects of a process go wrong or if the resulting product is unacceptable, the process being applied or the materials being produced must be adjusted. The **adjustment must be made as soon as possible to minimize further deviation**.

Everyone in the organization must ask this question regularly: *Are we better off than yesterday?*

Adaptation becomes more difficult when the people involved are not empowered or self-managing. A Scrum Team is expected to adapt the moment it learns anything new through inspection.

## Scrum Values

The Scrum values provide a code of behaviors, or ethics, for Scrum teams.

There is 5 Scrum values: FOCCR – Focus, Openness, Commitment, Courage and Respect



## Scrum Team

The Scrum Team consists of:

* **1 Product Owner (PO)**
* **1 Scrum Master (SM)**
* **Some Developers**

Here are some characteristics of a Scrum Team:

* Within a team, there are **no sub-teams or hierarchies**.
* A team must be **cross-functional**, meaning that they have all the skills necessary to finish each Sprint.
* A team must be **self-managing**, meaning that they internally decide who does what, when, and how.
* A team is **small enough to remain nimble and large enough to complete significant work within a Sprint**. The ideal team size is **7 members, ±2 (not counting the PO and SM)**. The researches indicate these numbers maximize people's ability to create close relationships and collaborate most effectively.

### Product Owner (PO)

PO is the business representative on the team. They're NOT part-time team members. They show up every day, because they're contributing to the final product every day.

Here is what a PO do:

* **Interact with the stakeholders** on a daily basis. Must be in tune with all the changes occurring in the business context. As a result, PO is the keeper of the product vision.
* **Define the backlog of work to be done** and the prioritization of those work items.
* Ensure the team members clearly understand the request details.
* Push the team to complete as much work as possible in each short delivery period.
* **Review all the work items** once finished. And either accepts it, or asks the team to make changes to ensure the highest value being delivered.

Who is stakeholder?

"Stakeholder" refers to people **outside the Scrum team who have an interest in the product** and can influence its development. Examples are very wide, such as end users, sponsors, direct managers, salespeople, legal officers, etc.

### Scrum Master (SM)

If you're wondering how on earth a team can keep up with PO’s demands, you're not alone. The Scrum founders recognize that they need to counterbalance the PO role, so they created the Scrum Master.

SM protects the team and the process. They ensure the team members are performing at a sustainable pace without being burned out before they reach the finish line. As a result, SM is the most visible spokesperson for the team.

Here is what a SM do:

* **Help all team members understand Scrum** theories and practices.
* Make charts and boards to **share the team's progress with anyone** (including stakeholders) who's curious or interested in how they're doing.
* Being the first escalation point when something gets in the way for the team. Work to **remove any impediments** until they're out of the way and the team can continue on.
* **Hold the team accountable for their commitments to the PO**. Show trends in team performance over time to help the team improve their processes and practices.
* Ensure that all Scrum events take place and are positive, productive, and kept within the timebox.
* Facilitate meetings and stakeholder collaboration as requested or needed

**In a nutshell, while PO focuses on what needs to be done, SM focuses on how the team does the work.**

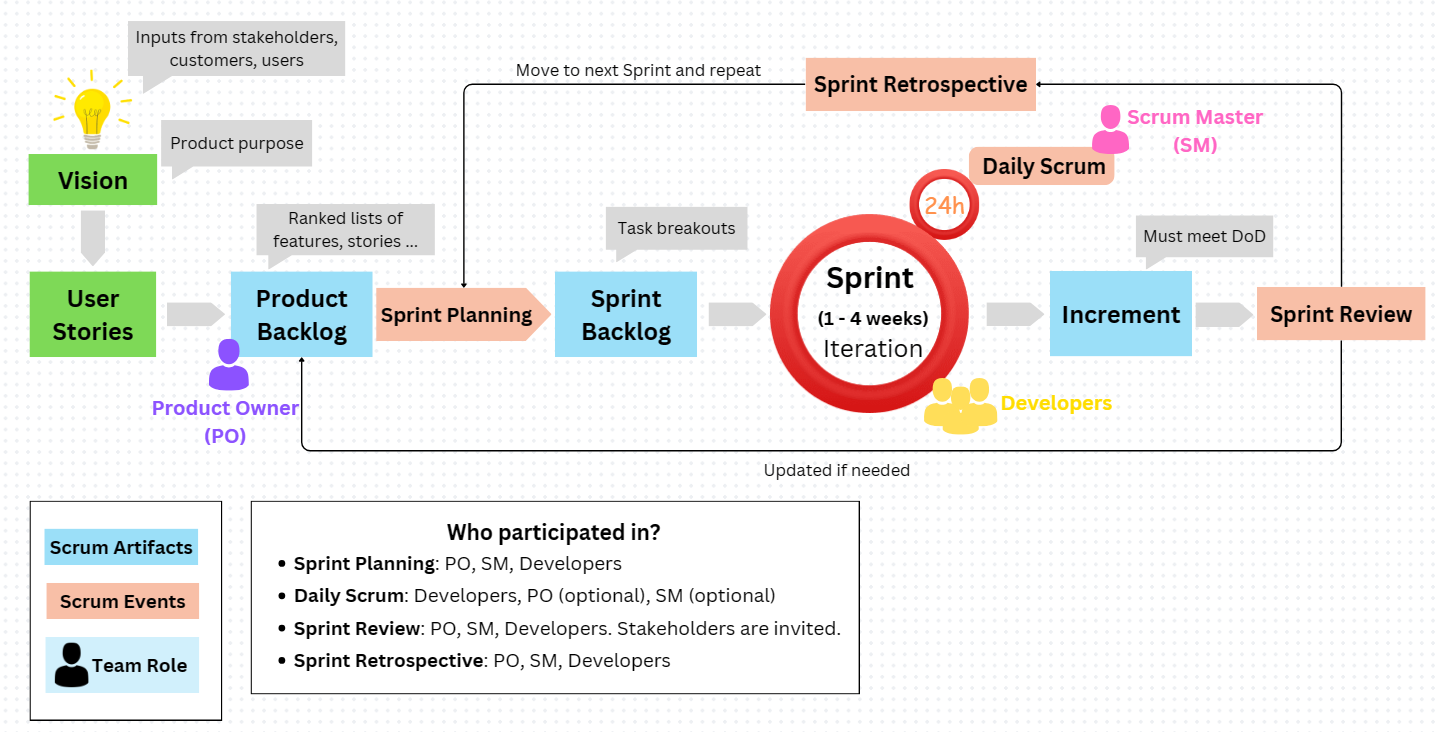
### Developers

Developers are any member of the Scrum Team who are committed to creating any aspect of an Increment in each Sprint. The PO and SM can be sometimes developers.

They are accountable for:

* Creating a plan for the Sprint, the Sprint Backlog
* Instilling quality by adhering to a Definition of Done
* Adapting their plan each day toward the Sprint Goal
* Holding each other accountable as professionals.

## Scrum Framework



# Scrum Preparations

## Project Vision

*You'd never go on vacation without choosing your destination and activities before you start my trip. Your Scrum project is exactly the same.*

The vision is the **map that will guide your team to your destination**. It tells everyone where they're going and what valuable product they'll deliver at the end. The PO establishes this before any work begins.

The destination contained within the vision is called a Minimum Viable Product (MVP). It’s about developing a product just enough to get out the door to early adopters. This approach is basically saying that if we do a small set of work to get our usable product out the door, we've met the goal.

There are some reasons for this:

* **Get fast feedback from stakeholders** and understand what else is needed.
* **Minimize the possibility of scope creep** or gold plating, where unnecessary things are added along the way.

## User Stories

Project vision is really useful to help us get organized, but it’s still too big for a team to work on and deliver value in small timeframes.

Each interaction customers or users have with our product is a use case that tells the team a story about how they're going to use our product. We call these User Stories.

Anyone on the team can write user stories, but it's usually the PO.

When writing user stories, it's best to use **INVEST as a guide**:

* **Independent**: Each user story should be **delivered separately from others**, and have value by itself. If you only have time to do one thing, this story can stand alone.
* **Negotiable**: Until the story is committed for work, it **can be changed or canceled at any time**.
* **Valuable**: The story delivers value to the PO, stakeholders and customers.
* **Estimable**: You must be able to estimate the size of the story in story points. That means that the story is **descriptive enough**. So, you know what has to be done to finish it.
* **Small**: The story is small enough to be **completed in one sprint**.
* **Testable**: The story provides **enough information to develop test cases** for it.

### How To Write User Stories?

**Format:**

Here is a simple format tells us so much: *As a …, I want to … so that …*

For example, a story for a mobile food ordering app could be:

* *As a mobile customer, I want to create a profile so that future orders are faster to place.*

This is known as a Functional User Story, meaning it serves a function for the end user.

Or another user story:

* *As a developer, I want to upgrade the database software to the latest version so that we have a supported product.*

This is known as a Non-Functional User Story, meaning it helps support the functionality the end users need without directly benefiting them.

**Acceptance Criteria (AC):**

Each user story should **has its own unique AC**. Your AC should be as explicit as possible. So all parties on the team know what done for the story really is.

In the case of our functional story about creating a profile, the AC could be:

* *Customer name is captured and saved.*
* *Customer email address is captured and saved.*
* *Customer phone number is captured.*
* *Saved and customer password is captured and saved.*
* *Invalid phone numbers, addresses and passwords are rejected*.

**Estimation:**

There are two kinds of estimation we use every day:

* **Actual estimation**: This is what you use when reading a map. It's 25 miles from point A to point B. Very specific!
* **Relative estimation**: This is comparing things to each other to get a general idea of something, like a giraffe is twice as tall as a zebra, or a cake is the same width as a pie.

In Agile and Scrum, we use both kinds of estimation.

* **Relative estimation**: To get a rough size of our work by comparing user stories to each other. Because user stories are rough statements of need, we can't be too specific on how big they’re. Relative estimating helps us maintain the mindset that it's **just an estimate, not a commitment**.
* **Actual estimation**: When we commit to tasks, we'd better be pretty clear on what it takes to do that task, so we need to be **specific on how long it will take**.

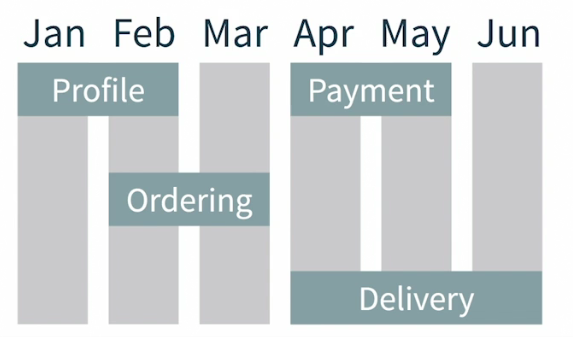
Estimation is meant to be lightweight and fast. It shouldn't take days, or even hours, to determine how much effort you'll put into something.

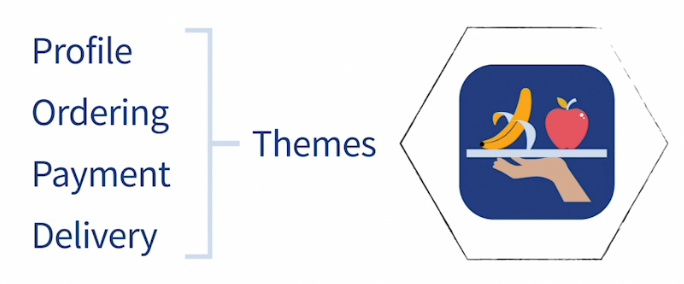
## Roadmap and Release Plan

You've know how long your Sprints are. Now it's time to **estimate WHEN things will be worked on**. Scrum has tools for this:

### Roadmap

With the eariler mobile food ordering app, we identified themes of *profile*, *order*, *payment*, and *delivery*. Now we need to **decide the best order to work on these themes**. While *payment* may be the most valuable to our business partners, from a dependency perspective it makes more sense to prioritize the *profile* and *order* themes first.

Here's what the roadmap would look like for these themes:



### Release Plan

The release plan **provides visibility to HOW we're going to deliver**. In Scrum, you must have fully-functional stories completed at the end of every Sprint. However, you're not required to release them at the end of every Sprint. This means you can complete all the work for 2-3 Sprints before releasing the combined results together.

You'll estimate the number of points to target for each Sprint. For example, we can handle about 10 points/2-week Sprint, and we want to release after 3 Sprints. That means about 30 points/release. Now let's take a look at our roadmap and stories we've organized over time based on our themes. All we need to do is select the highest-priority stories by size to fit into each Sprint. It should look something like this.



You might notice we're **not planning those in priority order**. At this level of detail, it's about maximizing the number of points per Sprint. So if the next highest priority story, Story C for 8 points, is too big for Sprint 1, we go down the list until we find one that will fit, Story D, 5 points.

Always remember that your roadmap is an estimate of when the team will complete these stories. The team will learn things and **write new stories throughout the Sprint, so the roadmap is meant to be updated after every Sprint if needed**.

# Scrum Events

## Sprint

Unlike the other Scrum events, Sprint **isn’t a meeting**. But it’s the **container for all of the User Stories** that should be done by a Scrum Team to achieve a Sprint Goal.

Each Sprint is a **heartbeat of Scrum** and consists of Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. Sprints have fixed length of **1-4 weeks**.

Every project is comprised of a series of Sprint. A new Sprint starts immediately after the conclusion of the previous Sprint. There are no time gaps between Sprints.

During the Sprint:

* No changes are made that would endanger the Sprint Goal.
* Quality does not decrease.
* The Product Backlog is refined as needed.
* Scope may be clarified and renegotiated with the Product Owner as more is learned.

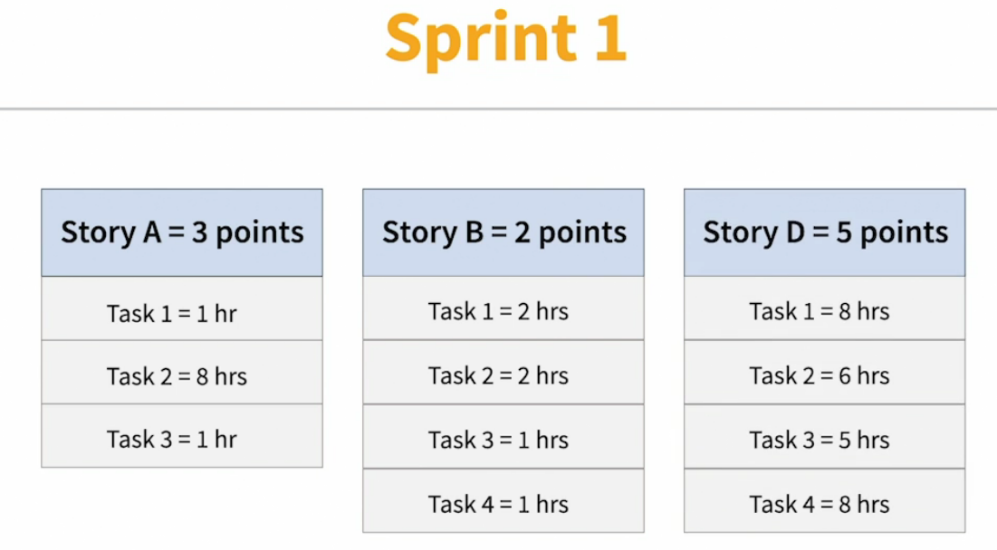
For tips on the Sprint, check <https://www.scrum.org/learning-series/sprint/making-your-sprints-more-effective>.

## Sprint Planning

Every Sprint starts with Sprint Planning where the whole team determines what they plan to accomplish during the Sprint.

Sprint Planning addresses the following topics:

* **WHY is this Sprint valuable?** The PO proposes how the product could increase its value and utility in the current Sprint. The whole team then collaborates to define a Sprint Goal that communicates why the Sprint is valuable to stakeholders. The Sprint Goal must be finalized prior to the end of Sprint Planning.
* **WHAT should be done in this Sprint?** Through discussion with the PO, the Developers select items from the Product Backlog to include in the current Sprint. The team may refine these items during this process, which increases understanding and confidence.
* **HOW will the chosen work get done?** This is often done by decomposing Product Backlog items into smaller work items of one day or less. How this is done is at the sole discretion of the Developers; no one else tells them how to turn Product Backlog items into Increments of value.
* **WHEN will this Sprint finished?** This is about estimating each Sprint by estimating each story points in the Sprint. For examples:



The Scrum Team makes this transparent by creating a Sprint Backlog which includes the Sprint Goal, selected Product Backlog Items and Developers’ plan for delivering the work.

Sprint Planning at a glance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Inspection** | **Adaptation** | **Participants** | **Timebox** |
| Sprint Planning | Product Backlog, Product Goal, Definition of Done | Sprint Backlog, Sprint Goal | Scrum Team | 8 hours for a 1 month Sprint |

For how to conduct Sprint Planning: <https://www.scrum.org/learning-series/sprint-planning/conducting-the-sprint-planning-event>

## Daily Scrum

A Daily Scrum is held by the SM. Here, the whole team get together for **15 minutes every day to share each member’s current status in order to prepare the upcomming day’s work**. It’s based on 3 questions:

* *What did you do since the last daily scrum?* (normally, yesterday)
* *What will you do today?*
* *What are impediments you having?*

**Note**: An impediment is something you CANNOT resolve yourself and need help from others. Thus, it's NOT an issue which is something you can resolve yourself with some extra time. For Risks vs Problems vs Issues vs Impediments, check [here](https://rolandwanner.com/risks-problems-issues-and-impediments-what-is-the-difference/).

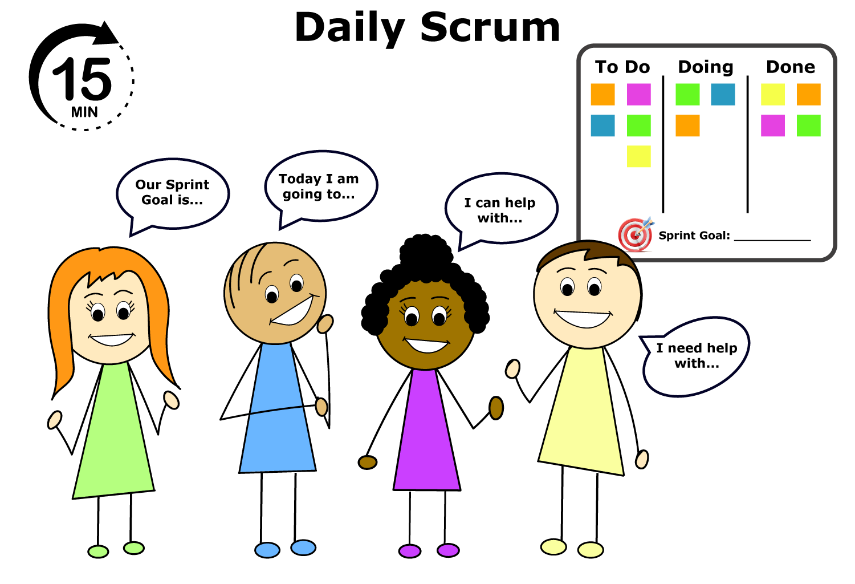
Daily Scrum at a glance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Inspection** | **Adaptation** | **Participants** | **Timebox** |
| Daily Scrum | Progress toward Sprint Goal | Sprint Backlog | Developers | 15 minutes |

The Daily Scrum helps:

* Ensure that all team members are aware of any updates/changes, and that, the team is making progress toward the Sprint Goal.
* Members ask for help when they need it and adjust the Sprint Backlog, if necessary.
* Promote quick decision making and may reduce the need for other meetings.

Note: The Daily Scrum is not the only time the team are allowed to adjust their plan. They often meet throughout the day for more detailed discussions about adapting or re-planning the rest of the Sprint’s work.



You might not know!

The name of this event is **NOT the Daily Stand-Up** or **Stand-Up Meeting**. Older versions of the Scrum Guide referred to this meeting as the "Daily Stand-up." However, it was changed to recognize that participants are not required to stand and in fact, some participants may not be able to stand during the meeting.

## Sprint Review

The Sprint Review is a working session where the whole team **presents their completed work to their stakeholders and asks for feedback and guidance**.

During the event, the team and stakeholders review what was accomplished in the Sprint and what has changed in their environment. Based on this information, attendees collaborate on what to do next. The Product Backlog may also be adjusted to meet new changes.

Sprint Review at a glance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Inspection** | **Adaptation** | **Participants** | **Timebox** |
| Sprint Review | Increment, Sprint, Product Backlog, Progress toward Product Goal | Product Backlog | Scrum Team, Stakeholders | 4 hours for a 1-month Sprint |

The Sprint Review is the second-to-last event of the Sprint and is timeboxed to a maximum of 4 hours for a 1-month Sprint. For shorter Sprints, the event is usually shorter.

For how to conduct Sprint Review: <https://www.scrum.org/learning-series/sprint-review/conducting-the-sprint-review>

## Sprint RetrospectiveSprint Retrospective

The Sprint Retrospective is the last event in a Sprint. It’s the time for the whole team to **review some aspects (like process, communication, skills, tools, etc.) in order to improve their effectiveness and how they work** as a team. Unlike other Scrum events where the focus is to improve the product, Sprint Retrospective is a place for the team to inspect and adapt their working practices.

The team typically discusses:

* What went well?
* What didn’t go well?
* What could be improved?

Sprint Retrospective at a glance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Inspection** | **Adaptation** | **Participants** | **Timebox** |
| Sprint Retrospective | Sprint, Definition of Done | Actionable improvements, Definition of Done | Scrum Team | 3 hours for a 1-month Sprint |

The Sprint Retrospective concludes the Sprint. It is timeboxed to a maximum of 3 hours for a 1-month Sprint. For shorter Sprints, the event is usually shorter.

For how to conduct a Sprint Retrospective: <https://www.scrum.org/learning-series/sprint-retrospective/conducting-the-sprint-retrospective>

# Scrum Artifacts

Scrum’s artifacts represent work or value. They are designed to maximize transparency of key information. Thus, everyone inspecting them has the same basis for adaptation.

Each artifact contains a commitment to ensure it provides information that enhances transparency and focus against which progress can be measured:

* For the Product Backlog, it is the Product Goal.
* For the Sprint Backlog, it is the Sprint Goal.
* For the Increment, it is the Definition of Done.

These commitments exist to reinforce empiricism and the Scrum values for the Scrum Team and their stakeholders.

## Product Backlog

The Product Backlog represents **what the Scrum Team needs to do in order to deliver the product**.



It consists of:

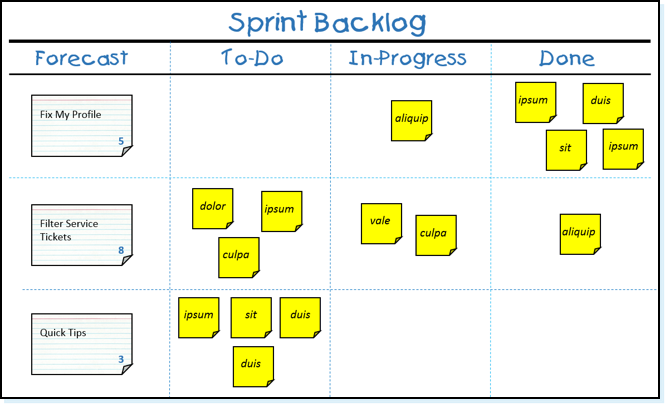
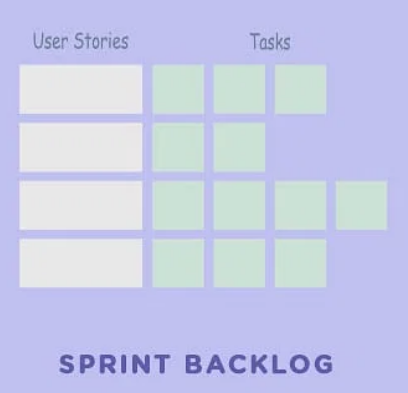
* **Product Backlog items (PBIs)** - each of which represents something that needs to be done.
* **A Product Goal** - that describes the Scrum Team’s current long-term objective for the product.

Product Backlog Refinement is the act of breaking down and further defining Product Backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order, and size. Attributes often vary with the domain of work.

As the work progresses, items in the backlog will be added, removed, reordered, and evolve on a continuous basis.

## Sprint Backlog

The Sprint Backlog represents **what the Developers plan to accomplish during the Sprint**. It describes the WHY, WHAT and HOW of the Sprint. It’s created by the Developers during the Sprint Planning event.



It consists of:

* **A Sprint Goal** - that describes the objective for the Sprint 🡪 WHY are we doing this Sprint?
* **A set of** **Product Backlog items (PBIs)** selected for the Sprint 🡪 WHICH PBIs will be addressed in this Sprint?
* **An actionable plan** for how the team will deliver the work (the Increment) and achieve the Sprint Goal 🡪 HOW will we achieve our Sprint Goal?

The Sprint Backlog is a plan by and for the Developers. It is a highly visible, real-time picture of the work that the Developers need to do. Consequently, the Sprint Backlog is **updated throughout the Sprint as more is learned**. It should have enough detail that they can inspect their progress in the Daily Scrum.

## Increment

The Increment is the **latest version of the product** **that conforms to the Definition of Done** (DoD). In other words: *an increment = work completed in the Sprint + all work completed in previous Sprints.*

What Is DoD?

A Increment is done when it conforms to stakeholder’s defined quality standards. These quality standards are known as DoD.

In scrum, when we say that something is done, we mean that it's fully done. **Doesn't need to be worked on again. Ready to be used by customers**.

During each Sprint, the Developers work toward the current Product Goal by implementing Product Backlog items (PBIs) and integrating their individual work together. The team and its stakeholders know that the product Increment is done, or usable, when it conforms to their defined quality standards. These quality standards are known as the Definition of Done.

As soon as the first Product Backlog item meets the DoD, the Scrum Team has created the first product Increment. If a Product Backlog item does not meet the Definition of Done, it cannot be released or even presented at the Sprint Review. Instead, it returns to the Product Backlog for future consideration.

Multiple Increments may be created within a Sprint. The sum of the Increments is presented at the Sprint Review thus supporting empiricism. However, an Increment may be delivered to stakeholders prior to the end of the Sprint. The Sprint Review should never be considered a gate to releasing value.

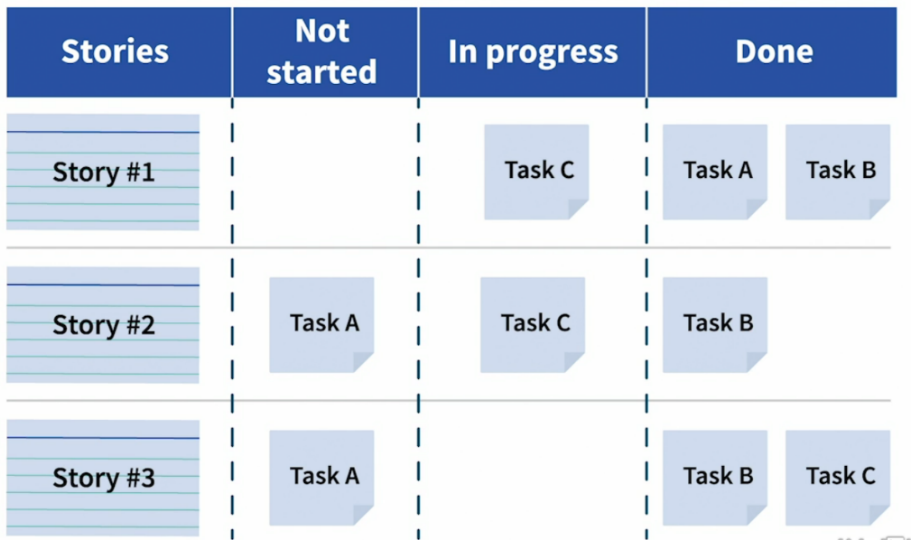
# Information Radiators

An information radiator is anything that helps everyone understand what they're doing and how it's going.

## Task Board

The task board can take any form you'd like, but it has a few key components: user stories, their tasks and their current status.

For example:

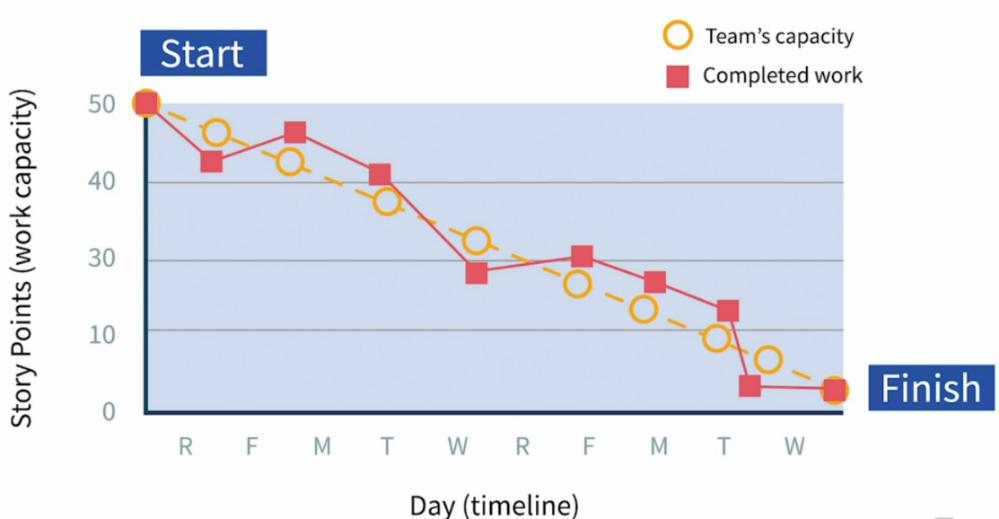


This board becomes the heart of your Sprint execution. The team will gather around it to track what's happening in their Sprint and help each other.

## Burn-Down Chart

It’s a chart which shows the **amount of work which is remained in a backlog**. It’s used by the team to measure how well they're executing the Sprint.

For example:

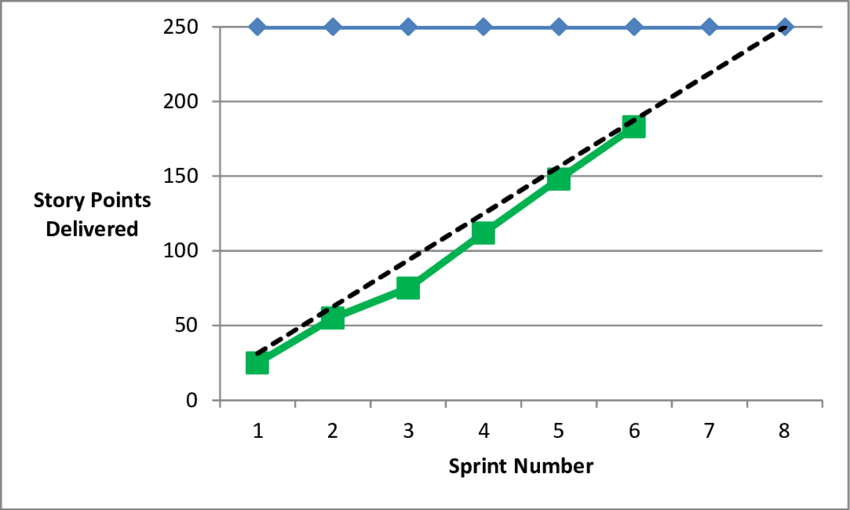


As you can see, the burndown tells you exactly where you’re in the Sprint by day and how much work you committed to do in the whole Sprint, or hours. Ideally, you'll burndown in a linear fashion. The actuals line tells you exactly how you're doing in comparison to the ideal.

## Burn-Up Chart

It’s a chart which shows the amount of work which has been completed.

For example:



# Scrum and DevOps

## Why Scrum Teams Need DevOps?

Building software involves close cooperation between 2 types of teams:

* **Developers and testers** who write and validate software. This is the "dev" part of DevOps.
* **Operators who help the software deployed and released to customers**. They must be good at servers, middleware, network, storage configurations, monitoring techniques, build process, automation tools, etc. This is the "ops" part of DevOps.



An Agile/Scrum team is a **cross-functional team**. Also, one of the key principles of Agile/Scrum is to **deliver working software** **frequently**, from a couple of weeks to a couple of months. So you need a culture of close cooperation between Dev and Ops to make your software delivery pipeline efficient. This is what DevOps attempts to achieve.