Memorize - presents - Agile-Scrum

1 What's Agile Scrum?

Agile **Scrum** (`Mêlée au rugby`) is one of the most used **Agile methodologies** (`méthodes agiles`). This is a kind of incremental method (wiki) that is repeating small cycles (**iterations**). A long time ago, the **V-Model** (`Cycle en V`) model (an extension of the Waterfall model) was the most used methodology. But, it **wasn't flexible**, the client often had to **wait a long time**, and the product **may not have been what he wanted**, so a lot of companies are moving to Agile methodologies.

As this is an incremental method, you will add functionalities incrementally. When your iteration ended, the client will check what you did, give you feedback, so that you know what to do in the next iteration.

Other Agile methodologies: XP (Extreme programming), Kanban.
The Agile manifesto (`Manifeste Agile`), signed in 2001, is defining the guidelines for Agile project: Agile Manifesto

2 Values and principles

Agile is a **mindset** (`état d'esprit`) 🤯, **Scrum** a framework 🛠️. Scrum is based on **3 pillars** (transparency, inspection, adaptation), and **5 values**

- **Courage**: face your problems, and do what must be done, **don't hide bugs under the carpet** (`pas de` 🐛 `sous le tapis`)
- **Focus**: concentrate on the sprint goal
- **Commitment**: be committed to your work
- **Respect**: be respectful to others
- **Openness**: be open to changes or evolutions

Agile methodologies are based on **12 principles** and **4 values** (principles and values (FR), principles (EN) and values (EN)). My summary is

- to do everything to make your client happy 😊
- allow changing the specifications ✍️
- frequently deliver a working software 💻
- make sure that users and devs are working together 😉
- always strive to do better 🥲

3 Scrum is involving 3 roles ...

Product Owner 🤓

The client interacts with the **Product owner**. He must understand the business/market requirements, and organize the tasks so that the client gets what he/she wanted.

He will redact (and update) the **Product backlog**. He may not be the only one creating entries in the backlog, but others should ask him/her before, as he/she should know everything inside the backlog.

Scrum master 📚

The **Scrum master** is **making sure that the developers** (the product owner, and the company too) **are only focusing on their job**, and not on applying Scrum. He should **organize (or sometimes facilitate) meetings**, **support the team**, **manage the tools**, do **reporting**, **deal with blockers**, and **"timeboxing"** (a meeting shouldn't last more than it was supposed to) so that the project is going smoothly.

The Scrum Master is an expert of Scrum, and he/she must **ensure that Scrum is used properly**, but if the team agree on changing something, he/she must be **open for changes** (as he/she follows the principles and values of Scrum).

Development team 🚀

They are the people doing the work. This is usually a team of 3 to 9 persons, but some are adapting Scrum for bigger teams. **Notice** that is **Development**, **not Developers**. You can have **graphics/UX designers**, **sound engineers**, **artists**, or **testers**, along with the developers. Anyway, we are considering all of them as **developers** in the end, so you can use both.

4 ... and 4 ceremonies.

The development is made of iterations, a repetition of cycles, called **sprints** (=**iteration**). The result of your sprint is usually called **increment**. A sprint is made of **4 ceremonies** (`rituels`).

1. **Sprint Planning**: when starting a sprint
2. **Daily Scrum**: daily meeting
3. **Sprint review**: at the end, demonstration to the client
4. **Sprint retrospective**: usually at the end, look for improvements

Usually, a sprint lasts **2 to 4 weeks**. Once you picked a duration, you shouldn't change it (unless you got a good reason).

A sprint is considered as a project as a whole, so you have to write tests and the documentation during the sprint. You will do the same as you are doing for the project and add functionalities one after another until you reach your goal.

4.1 Sprint Planning

The main goal of the **Sprint Planning** is to decide your **Sprint goal**. Your **Sprint goal** is what you want to make at the end of the sprint.

- **Involves**: PO, SM, Dev. team
- **Duration**: up to 4 hours/4 weeks

The Product Owner is **proposing a goal**, and the **items** of the **product backlog** that you may do. The dev team is picking items (=>value for the client, effort), explicitly defining when they are considered to be "done" (doc? tests?), in the **sprint backlog**. You need to have a clear idea of **what makes the sprint successful**, and how you will achieve the **sprint goal**.

You will make **two of Scrum's artifacts**

- Product backlog
- Sprint backlog

4.1.1 Product backlog

The **Product backlog**, is a sort of project specification (`cahier des charges`). This is a list of tasks. You must add the features (~=**user stories**) that the client requested.

- Usually, they are **prioritized** (MoSCoW)
- Usually, they are sorted by **importance** (most important to less important)
- Usually, you must add the dependencies between tasks

> **Note**: reviewing the backlog is sometimes called backlog **grooming** (or refinement)
> **Note**: Usually, the Product Owner and the Scrum Master are preparing the Product backlog **before** the meeting.

4.1.1.1 User Stories and Epics

An **Epic** is a group of **(user) stories** that are "similar" (ex: all stories related to the user management). A User Story is a text describing a task (~= requirement) from the point of view of a user. You got stories about

- **features** (functional)
- **bugs** (technical)
- **tests** (technical)
- **technical debt** (technical): this is what you have to do before starting to improve the code (refactor/create constants/normalize things/...)
- **actions** (technical): changing your way of doing things (=from a retrospective)

A user story is supposed to provide content about the task, so that you are "working with" the one that actually needs it. We must know the **reason**, the **goal**, the **value**, and an **estimation** of the cost/value for the person needing this task to be done.

The usual template is `"As a <type of user>, I <want to do something/goal>, so that <a reason/benefit>."`.

> **TIP**: give colors according to the kind of stories. Or, you may give colors to epics and symbols to stories.
> **Note**: You can see an epic as the result of a succession of stories, and a group of epics is making an initiative (omitted).

4.1.1.2 Priority

You must filter the tasks by priorities, most likely using the **MoSCoW** notation.

- **M**: Must have this
- **S**: Should have this
- **C**: Could have this
- **W**: Would have this

You may ask the client about the priority and/or deduct it by yourself, since some tasks that the client wants may be too hard/useless if done in the first increment. You must take into account **the value for the client**.

4.1.1.3 Estimation

You need to evaluate how much **complex** a task is (or how much **effort** it will require). We are not evaluating the duration like in non-agile projects.

- You may **compare tasks** and sort them by difficulty.
- You can give a value among a fixed list (`0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?, inf.`) to each task. The average/median value may be used as an answer.

If you got outliers (`valeurs abérantes`) such as (almost) everyone gave "10", one gave "5", and one gave "20", then it would be a good idea to ask them **why** they gave this value (maybe they through of something others didn't).

- The T-Shirt Size Estimation

> You should learn from your previous estimates. You may pick a previous task and use it as a reference.

4.1.1.4 The backlog itself

You may create a table, in which each row is a story. You may group them by epics if you want. As for the columns, here are some ideas

- # or ID: ID of the story
- Title or Title+Description: the story
- The priority
- The importance, estimated with
  - An estimation of the complexity
  - An estimation of the value for the client
- Some notes
- The Sprint: the one in which you completed this story (empty if you didn't)
- The Epic: if this story belongs to an epic
- The dependencies

4.1.2 Sprint backlog (1/2)

The **Sprint backlog** is a list of tasks that the **dev team** took for the product backlog. They are the **tasks that they are planning to do** during the sprint. The tasks are **split into small tasks**, taking **a few hours and up to one day** to be **"done"**.

The developers are taking tasks according to their **capacity**. This is the sum of the **evaluation/score** of each task you picked in the last sprint.

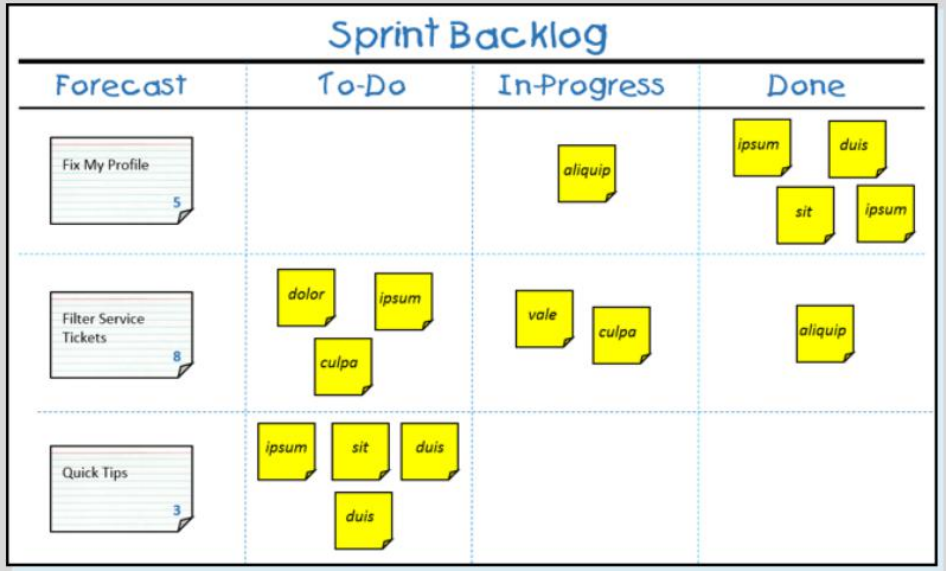
> **Note**: do not take too many "unknown-risk" stories. You should also take the previous sum of your estimations when picking tasks.

We are using the famous **whiteboard**, with the following columns

- **Todo** (`À faire`): what has to be done
- **In-progress** (`En cours`): working on it
- **Review** (`revue/relecture`): optional, someone else is checking what you did
- **Tests** (`Tests`): optional, testing the product
- **Documented** (`Documentation`) :optional, working on the documentation
- **Done** (`Terminé/Fini`): done
- **Canceled/Blocked** (`Terminé/Fini`): optional, you won't continue a task, or you can't

Your goal is to move all of your cards in **Done**. You should add on each card the person that will do each task.

4.1.2 Sprint backlog (2/2)



We are usually using online tools such as

- Trello (Atlassian/Jira)
- GitHub (Microsoft)
- YouTrack (JetBrains)

4.1.2 Scrum release planning

A **release plan** is summarizing a lot of information about a sprint, so you may use one too. This is also providing information for every sprint, the **roadmap** of your project.

- duration of a sprint
- deadlines
- budget
- sprint goal
- **Acceptance criteria**: What's the definition of done
  - tests? (coverage? deployment to a production environment?)
  - doc?
- Guidelines/Guidance to write better code/improve quality

4.2 Daily Scrum or standup/stand-up

- **Involves**: SM, Dev. team, PO?
- **Duration**: up to **15 minutes**

Check the **progress**. What each one did **yesterday**, what are they planning to do **today**. Identify **blockers** and **challenges**. The purpose is **informative**, a sort of coffee break (`pause café`). If you don't drink coffee, maybe chatting while playing a game might be a new way of approaching this (ball, cards, etc.). Everyone is supposed to stand up (`rester debout`).

- You can use **Round Robin**: everyone is answering three questions (yesterday? today? blockers?)
- You can use the **Board-based** method (**walking the board**): simply ask easy one about their task, starting from the **right** to the left
- Some are using a token to the one talking (ex: ball), the person talking will throw the ball to the next person (not a neighbor)

My Advice

> **As a Scrum master**: Try to relay as much as possible information in your group. It may be a good idea to have a **one-to-one talk** with each member, once a day. This may help them think of new ways of doing something, help them writing documentation/thinking of new tests, and broaden their horizons (and yours too).

4.3 Sprint review

- **Involves**: PO, SM, Dev. team, the client/users
- **Duration**: up to 4 hours/4 weeks

At the end, you must **demonstrate** to the client (and your team sometimes) **what you did**, the items you completed (also talk about what you planned, but didn't made). Take note of what the client wants to **review** in **the product backlog**.

Each developer may **demonstrate** what they did (and why?), but do not make it too long, like 5 minutes per person. Do not forget to talk about **key metrics**: you are making a software/website/... for customers, check that it is usable/...

> Some organizations are making weekly demo with the team, so that everyone knows what the others are doing, understand a bit more what they need to do, and keep track of what the client wants.

4.4 Sprint retrospective

- **Involves**: PO, SM, Dev. team
- **Duration**: up to 3 hours/4 weeks

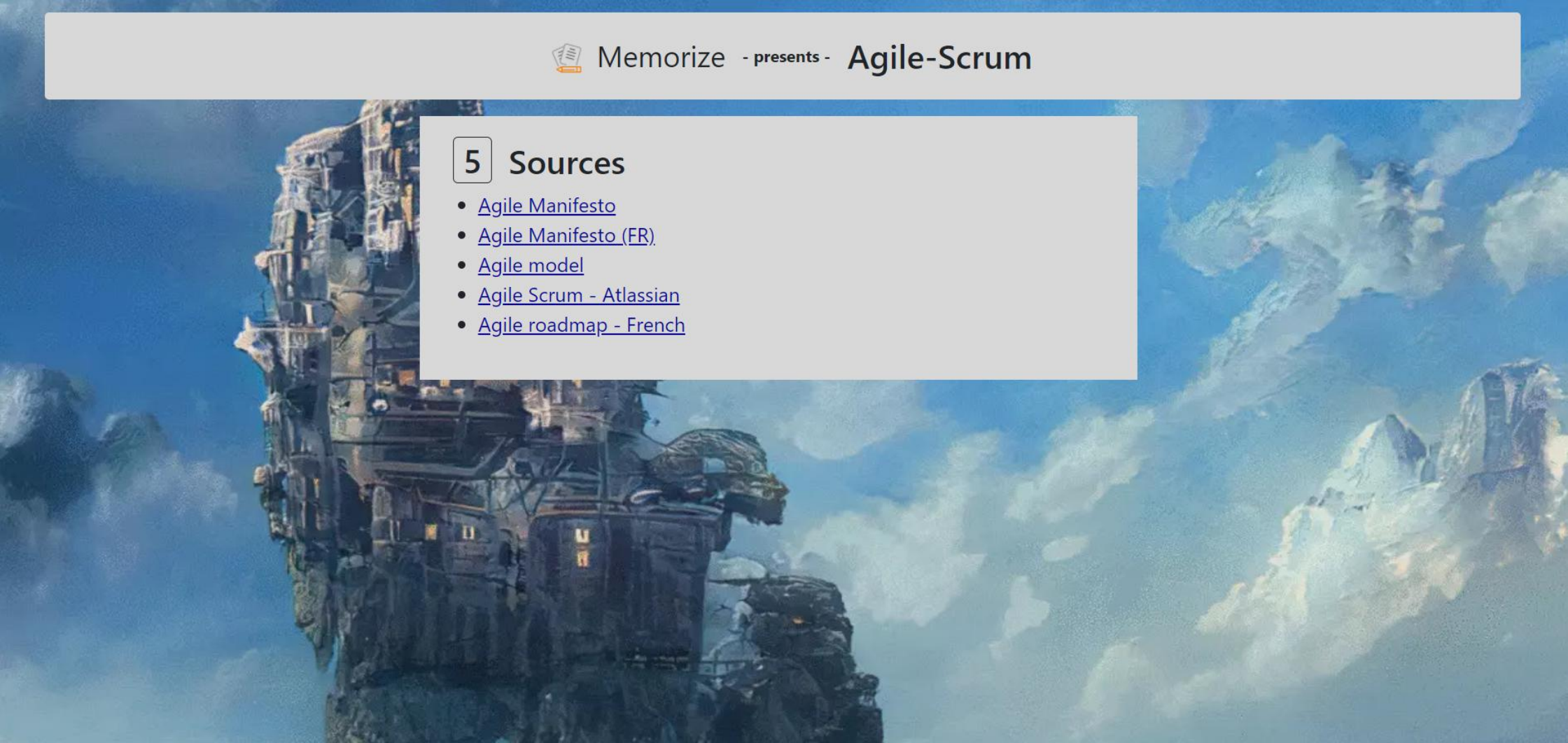
At the end, identify the areas of improvement. Find **what didn't work well** and of course **what worked well**.

- tools (`outils/logiciels`)
- peoples
- directives (`règles: écriture du code, comportement, etc.`)
- processes (`organisation`)
- ...

At the end, you must select **one thing** and focus on it. Focusing on one is making sure that I least one thing will improve per sprint. You might write an **action plan** of what to do.

Note: You can do this retrospective in the middle of a sprint.

Memorize - presents - **Agile-Scrum**



5 Sources

- Agile Manifesto
- Agile Manifesto (FR)
- Agile model
- Agile Scrum - Atlassian
- Agile roadmap - French