
Agile Methodologies

Building upon the foundational principles of Kaizen and the structured approach of the PDCA cycle, the software development industry has evolved its own set of iterative methodologies tailored to the unique challenges of creating digital products. A pivotal moment in 2001 was the creation of **the Agile Manifesto**. This document encapsulates **a set of values and principles to guide an iterative and people-centric approach to software development**.

To better understand how the core values translate into real-world practices, let's compare their practical application in agile approach with traditional corporate approach.

1. ***Individuals and interactions over processes and tools***

Traditional Corporate Approach:

- *Rigid processes and tools*
- *Clearly defined roles and responsibilities*
- *Emphasis on following established procedures*

Agile Approach:

- *Focus on individuals and interactions*
- *Flexibility and adaptability*
- *Enhanced communication between developers, customers, partners, and colleagues*

First value is individuals and interactions over processes and tools. The focus should be on enhancing the interactions between people developing the SaaS product and their customers, partners, and colleagues. They should be given the freedom to react and interact with the environment. In contrast, corporate environments often prioritize processes and tools to ensure standardization and minimize mistakes, especially when a company has been producing a successful product. As you see, this shouldn't be true for software as a service developers. They should try to enhance individual interaction.

2. ***Working software over comprehensive documentation***

Traditional Corporate Approach:

- *Comprehensive documentation*
- *Extensive planning before implementation*
- *Long development cycles*

Agile Approach:

- *Emphasizes working software*
- *Rapid development and deployment*
- *Creating only necessary documentation*

Secondly, there is working software over comprehensive documentation. If you have experience as a programmer, you know that there are two main components: the actual working software which is the code and the documentation that describes that code, making it easier for others to understand and use. For software as a service developers, the emphasis should be on the first part – the working software. Developers must ensure that this aspect of the development process is functional and effective because software as a service teams are typically small. They are likely to collaborate closely and work together on the code itself. In this context, extensive documentation may not be as necessary and could even be considered a waste of time. On the other hand, in the corporate world, documentation is more important. There may be hundreds of people involved in the development of a product, and the process could span several years. Teams are likely to change over time, making comprehensive documentation essential for maintaining continuity and understanding. While working documentation is valuable in the corporate setting, it may not be as critical for software as a service developers, given their smaller team sizes and more focused development efforts.

3. ***Customer collaboration over contract negotiation***

Traditional Corporate Approach:

- *Contract negotiation*
- *Clearly defined terms and conditions*
- *Maintaining existing customer relationships*

Agile Approach:

- *Prioritizing customer collaboration*
- *Building relationships for feedback and references*
- *Adapting the product based on customer input*

Then we have customer collaboration over contract negotiation. Software as a service developers should prioritize customer collaboration, not just for revenue but also for the valuable feedback and references they provide. Collaborating customers are more

important than actual revenues in this context. In the corporate world, contract negotiation is more critical due to the company's established position and the importance of setting clear terms and conditions for product sales.

4. ***Responding to change over following a plan***

Traditional Corporate Approach:

- *Follow rigid plans*
- *Resist change to minimize risk*
- *Stick to what has proven successful in the past*

Agile Approach:

- *Responding to change*
- *Adapt quickly to market shifts and user needs*
- *Embrace flexibility as a core strength*

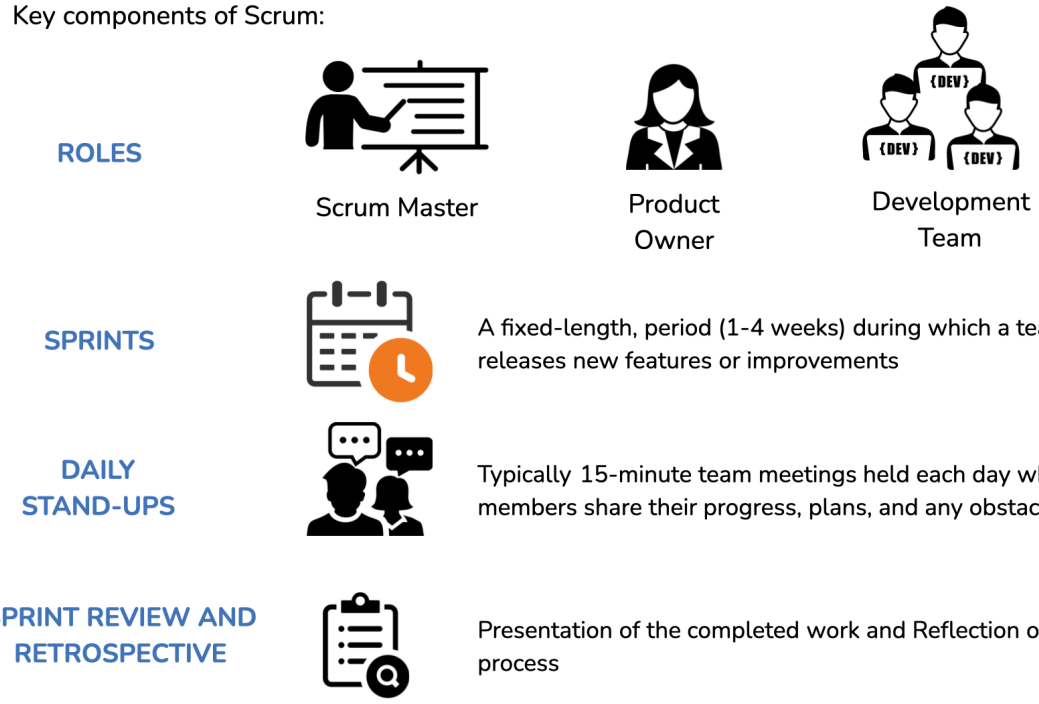
The last point is maybe the most important, responding to change over following a plan. The software as a service market is highly dynamic, with things changing rapidly on a daily basis. In this environment, it's crucial for your company to be able to react and adapt to these ongoing changes. In contrast, well-established corporate companies often have a plan that employees are expected to follow rigidly to minimize the risk of failure. This approach makes sense for them, as it helps ensure that nothing goes wrong. When a company has already proven that something works successfully, they are likely to stick with it. In this context, rigidity has its justification. However, software as a service companies should prioritize flexibility instead. The ability to be agile and responsive to market changes is essential for success in the constantly evolving software as a service industry.

The Agile Manifesto is significant not only for its key principles but also for the concrete methodologies it has inspired. We will explore Scrum, Kanban and Extreme Programming.

Scrum

Scrum is one of the most popular Agile frameworks, widely used for its **structured approach to managing complex projects**. It's particularly well-suited for SaaS projects due to its emphasis on flexibility and regular delivery of working software.

Let's examine the key components of Scrum:



- ***Roles**: Three primary roles - Scrum Master, Product Owner and Development Team*

Firstly, Scrum defines three primary roles, each with distinct responsibilities: **Scrum Master, Product Owner, and Development Team**. The Scrum Master facilitates the Scrum process, removes impediments, and helps the team adhere to Scrum principles and practices. The Product Owner acts as the voice of the customer, manages the product backlog, and ensures the team delivers maximum value. And the development team represents a cross-functional group responsible for delivering the product increment at the end of each sprint.

- ***Sprints:** A fixed-length, period (1-4 weeks) during which a team release new features or improvements*

Then, there are sprints typically lasting 1-4 weeks. During each sprint, the team works to create a potentially shippable product increment. This rapid cycle enables SaaS teams to frequently release new features or improvements, gathering user feedback quickly.

- ***Daily Stand-ups:** Typically 15-minute team meetings held each day where members share their progress, plans, and any obstacles*

Next, we have daily stand-ups which are short, daily meetings. During these meetings, team members briefly share what they've accomplished, what they plan to do next, and any obstacles they're facing. This regular communication helps identify and address issues promptly, maintaining the project's momentum.

- ***Sprint Review and Retrospective:*** *Presentation of the completed work and Reflection on the process*

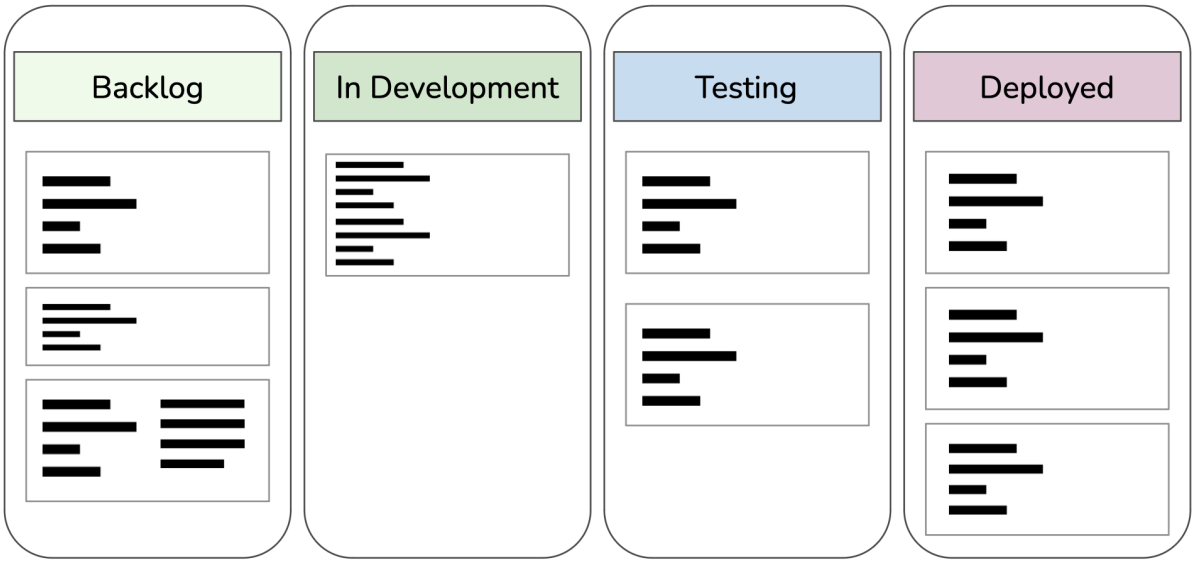
And finally, at the end of each sprint, there are two events occurring. During Sprint Review, the team presents the completed work to stakeholders, including the Product Owner and potentially customers or end-users. This is an opportunity to demonstrate new features or improvements, gather feedback, and ensure the product is evolving in line with user needs. In Sprint Retrospective, the team reflects on their process, identifying what went well and what could be improved. This continuous self-assessment allows the team to refine their practices, boost efficiency, and address any recurring issues.

Kanban*: Visualizing workflow and limiting work in progress to improve efficiency.*

Next, let's explore the methodology called Kanban. It is a visual-based Agile methodology that focuses on managing and optimizing workflow. Unlike Scrum, Kanban does not use time-boxed iterations. Instead, it emphasizes continuous delivery and improvement.

Let's explore its key components:

- ***Visual Board***



The heart of Kanban is its visual board, which provides a clear overview of the entire workflow. A typical Kanban board might have columns representing different stages of development, such as:

- Backlog
- In Development
- Testing

- and Deployed

This visual representation allows team members to quickly understand the status of various tasks. Unlike Scrum's sprint-based approach, Kanban allows for a continuous flow of work. Tasks move through the board as they're completed, without waiting for a sprint to end. This flexibility is invaluable in SaaS environments, where priorities can shift rapidly based on user feedback or market trends. Work is "pulled" into the next stage when there's capacity, rather than being "pushed" according to a predetermined schedule. This approach helps prevent bottlenecks and ensures that each stage of the process is operating at optimal capacity.

- ***Work-In-Progress (WIP) Limits***

One of Kanban's most powerful features is the use of Work-In-Progress limits, or WIP. These caps on the number of tasks allowed in each stage of the process prevent overloading the team and maintain a steady flow of work. For example, if the "In Development" column has a WIP limit of 5, no new tasks can be moved into this stage until one is completed and moved to "Testing." These limits help teams focus on completing high-priority tasks rather than starting too many at once.

Extreme Programming (XP)*: Stresses technical excellence and customer satisfaction through practices like pair programming and test-driven development.*

And lastly, let's learn about Extreme Programming, often abbreviated as XP. It is an Agile methodology that emphasizes **technical excellence and responsiveness to customer needs**. Despite its "extreme" name, Extreme Programming is all about **simplicity** and **communication**.

Now let's understand the core principles of this methodology:

- ***Frequent Releases:*** *Regularly deliver small, working pieces of software to quickly get new features or improvements into users' hands.*

Firstly, there are frequent releases. Extreme programming promotes releasing small, working pieces of software often - sometimes even daily. For SaaS products, this means getting new features or improvements into users' hands quickly.

- ***Simple Design:*** *Implement the simplest solution that meets current needs, avoiding over-engineering and facilitating easier changes as requirements evolve.*

Extreme Programming promotes the simplest solution that meets the current needs. This principle helps SaaS teams avoid over-engineering and allows for easier changes as requirements evolve. For example, when adding a search function to a SaaS project

management tool, the team might start with a basic keyword search before considering more complex features like filters or advanced search operators.

- ***Test-Driven Development (TDD):*** *Write automated tests before coding to ensure robust features and reduce bugs in SaaS products.*

Then we have Test-Drive Development, or TDD for short. Extreme programming emphasizes writing automated tests before the actual code. For SaaS, this practice ensures that features are robust and reduces the chances of bugs when the product scales or when new features are added.

- ***Pair Programming:*** *Two developers working together, enhancing code quality and knowledge sharing.*

Next, the developers work in pairs, one writing the code while the other reviews it in real time. This practice enhances code quality, encourages knowledge sharing, and reduces the likelihood of defects.

- ***Continuous Integration (CI):*** *Frequently integrate code changes into a shared repository with automated testing to catch issues early.*

Extreme Programming also promotes frequent integration of code changes into a shared repository, where automated tests are run to detect issues early. This is critical for SaaS products, because it enables teams to release updates frequently and reliably without breaking existing functionality.

- ***On-Site Customer:*** *Maintain constant communication with a customer representative to answer questions and set priorities.*

And finally, this methodology suggests having a customer representative available to the team at all times to answer questions and set priorities. In the SaaS world, this often translates to close collaboration with product managers or UX researchers who have direct insight into user needs.

Each of these Agile methodologies – Scrum, Kanban, and Extreme Programming – offers unique strategies for implementing Agile principles. Scrum provides a structured approach with regular intervals for review and adjustment. Kanban offers flexibility and a continuous flow of work, which is ideal for managing ongoing tasks and unexpected changes. Extreme Programming focuses on technical excellence and customer satisfaction, ensuring that high-quality code is delivered consistently.