
AGILE
DEVELOPMENT

Table of Contents

Table of Contents

Introduction

Welcome to the Overview on Agile Development, where most of the confusing and daunting information on Agile Development is made easier to understand. This overview is meant to explain the Agile Movement as a whole and showcase some of the best information on the web regarding Agile. In here you can find the best practices, vocabulary terms, popular frameworks and more on the Agile Movement.

Agile

What is the Agile Process?

“Agile is not just a methodology, but a set of principles and philosophy” - Pearl Zhu

Many of the Agile Movement ideas surfaced around the 1970s as a reaction to traditional approaches to project development. But the Agile Development Movement was first introduced to the world in February 2001, when a group of 17 software developers met in Utah to discuss lightweight development methods. Together the seventeen published the Manifesto for Agile Software Development, in which they shared their combined experiences of software developing and helping others to do it.

The Agile Process is in turn a collection of principles and methodologies based on iterative software development, where requirements and solutions evolve through collaboration. While the Agile Process is to be considered as the fundamental principle behind the Agile movement, there are different ways to practice these principles and turn them to concrete examples, some of these are widely popular like Scrum and Kanban, while others are less known like XP and Lean.

To set the pace for this overview just remember Agile is not a Methodology! rather it is a movement that seeks alternative project management approaches to help teams to help teams respond to unpredictability through incremental, iterative work cadences and empirical feedback. Now you probably want to know what are the benefits and whether or not it is for you?

The Agile Benefits

Let's be honest, we all wanna know what makes Agile so enticing? Is it the fact that so many people use it? Is it because you won't need to do Waterfall anymore? Or is it simply because it sounds cool?

All of those are part of the answer, but in truth there's a lot of reasons as to why Agile is enticing and beneficial to individuals and teams. Let's shed some light on the most important benefits of Agile Development:

- Better decision making through regular team meetings and project direction.
- Cost reductions by keeping both the team and stakeholders up to date, which reduces risks of unsatisfied customers and additional work.
- Transparency between team members, stakeholders, project managers, and all other involved parties.
- Preparedness for unknown variables by focusing on smaller tasks instead of bigger ones.
- Safer revisions by maintaining regular project meetings and realizing tests through every iteration.
- Increases overall team satisfaction by splitting projects into smaller tasks which in turn reduce misunderstandings, potential impediments and stressful deadline scenarios.
- Promotes a team environment through which members get to know each others strengths and weaknesses in order to overcome them together.

These are a few of the more important benefits, but the list of benefits is much longer than this. Now that we covered those, you are probably wondering whether it's for you or not?

Is Agile for You?

Figuring out whether the Agile Process is for you can be difficult, if not daunting at times. But that is what I am here to do, help you figure out whether or not it is for you. Let's start by jotting down who uses it and who doesn't, then from that you can decide whether it is appropriate for your project or not.

Who uses the Agile Process?

The question of who uses Agile can be boiled down to various development teams, but let's cover some of the more common ones:

- Teams with projects that evolve over time
- Developers who want a more reactive approach to development
- Teams that work with stakeholders and outside sources
- Developers that need a more organized work environment
- Teams that want a process from which they can learn and improve dynamically
- Teams who have cross-functional skills
- Teams looking to work more collaboratively
- Teams who want more interactivity between the client and the team
- Developers that prepare for unknown variables

These are the most common groups of people who utilize the Agile Process. But this doesn't mean that it is designed to work for every team or project.

Who doesn't use the Agile Process?

There are many people willing to have heated discussions on why Agile doesn't work, as well as people who simply do not think it fits there project. The truth of the matter is Agile may not work for some projects or with some people. Some of those people are:

- Developers who do not work on interpersonal skills
- Teams dedicated to fixing issues and bugs
- Developers that only practice code in coding languages and not in plain english
- Teams with projects that are on a huge scale right from the start
- Developers who do not practice iterative software development
- Businesses who prefer keeping the project behind the scenes until release
- Teams that prefer to have a management position over the project
- Developers that prefer to work independently
- Teams that have already invested time into learning a different project management process like Waterfall or Spiral

Same as the people who use it, these are the most common groups that do not utilize the Agile Process. Many have tried it and realized it was just not the right fit, and many have not tried it but decided that it is the wrong way to approach project management.

What about you?

In the end the Agile Process falls into the category of a development tool. It can be beneficial at times, while it can also be detrimental during others. In order to make the most out of this tool you have to learn about it and what is the best way to apply it to your project. Thus the final decision on whether or not Agile is for you, is ironically up to you.

Terms to Know

These terms are used in various Agile frameworks and are crucial to learn when undertaking an Agile approach in a project or task. Most of these terms or practices are used in all Agile frameworks, but some are specific to one framework. Once they are learned you are a step closer to being a professional Agile Developer and understanding every Agile Process.

- **Agile Software Management** - The project management approach of developing increments of software in frequent iterations based on evolving requirements.
- **Backlog** - Is a changing list of product requirements based on the customer's needs. The backlog is not to be confused with a To-Do list, rather a list of desired features.
- **Burndown Chart** - A burndown chart represents all outstanding work, where the vertical axis represents the backlog, while the horizontal axis represents time. Burndown charts are used to measure the progress of an agile project at both a iteration and project level. The work remaining can be represented by story points, ideal days, team days, or other metrics.
- **Daily Standup/Meeting** - is a key component of Agile methodologies and serves as a daily forum for Agile teams to share progress, report impediments, and make commitments for the current iteration or sprint. This brief, 15-minute meeting is usually held every morning at the same time and in the same location.
- **Epic** - A very large user story that, in their current state, would be difficult to estimate or to complete in a single iteration. They are typically lower priority and are waiting to be broken down into smaller stories.
- **Fail-Fast** - Is the process of starting work on a task or project, obtaining immediate feedback, and then whether to continue working on the task or take a different approach.
- **Iteration** - Is a fixed or timeboxed period of time, generally spanning two to four weeks, during which an Agile team develops a deliverable, potentially shippable product.
- **Kanban** - Is a highly visual framework that uses continuous workflow rather than fixed iterations to produce shippable deliverables. Kanban focuses on completing entire projects rather than sprints.

.

- **Planning Poker** - Also called scrum poker, is a team-building and consensus-based exercise or game used for estimating workloads.

- **Product Owner** - Represents the voice of the customer and is accountable for ensuring that the team delivers value to the business. The product Owner writes the acceptance criteria, prioritizes them, and then adds them to the product backlog. Scrums teams typically have one Product Owner.

- **Scrum** - Is a popular framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is based on the adaptive and iterative methodology of software development that follows a set of predefined roles, responsibilities and meetings.

- **Scrum Master** - Is not the team leader but is often viewed as the team coach who acts as a buffer between the team and any distracting influences. The Scrum Master ensures that the Scrum process is used as intended, they have no authority over teammates, but due have complete authority over the process. A key part of their role is to protect the team and keep them focused on the tasks at hand.

- **Sprint** - A sprint is fixed-length iterations during which one user story or product backlog item (PBI) is transformed into a potentially shippable deliverable. Each sprint is assigned a set amount of time to be accomplished, which is usually between one week to one month.

- **Stakeholder** - Loosely refers to anyone outside the Scrum Team who has an interest in the product that the team is producing. Stakeholders can include but are not limited to direct managers, subject matter experts, account managers, salespeople, and legal officers.

- **Story Points** - Non-unit of estimation measuring complexity and size. Story points are relative, not absolute, and do not relate to actual hours.

- **Task** - Is a single unit of work broken down from a user story. Tasks are estimated in hours (or story points) remaining by the developer working on them.

- **Task Board** - An Agile task board is a physical or online visual representation of user stories broken down into tasks or work units.

- **Technical Debt** - Refers to the obligation a development team incurs when they use a short-term, expedient approach to developing software package without considering the long-term consequences. It increases project costs and complexity due to issues introduced through the software package.

- **Test Driven Development (TDD)** - TDD is the practice of designing and building tests for functional, working code, and then building code that will pass those tests.

- **User Stories** - A user story is a very high-level definition of a requirement written from the end-users point of view, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

- **Velocity** - It is a relative number which describes how much work the team can get done per sprint.

Popular Agile Frameworks

Remember at the beginning when I mentioned how Agile was a collection of principles and rules set to increase productivity and decrease risks. Well how do we turn these principles and rules into a system that we can follow? Not to worry there are many frameworks of Agile in which these rules and principles are set in motion for project teams to use.

In order to learn about these from an overview point of view I will cover some of the more popular frameworks by giving you some context behind how they work and can be applied to projects.

The Scrum Way

Scrum is the most widely used agile framework, which implements incremental product development by using one or more cross-functional, self-organizing teams of about seven people each. It provides a structure of roles, meetings, rules and artifacts. Scrum uses fixed iterations of time commonly referred to as sprints, during which teams tackle story points for their project goals.

Each team is responsible for providing potential deliverables at the end of each sprint, which typically lasts anywhere between one weeks to one month. During this sprints Scrum Teams hold various meetings in order to keep all team members updated on all progress, impediments, accomplishments and tasks.

Each teams can have multiple roles, but they all have one specific Scrum Master, which is not a manager, but a leader and coach for the rest of the team. The Scrum Master is held responsible for keeping the team on track and making sure that all team members know their respective roles and tasks.

The Scrum approach is more transparent than most frameworks where the team is in constant communication with the product owner and stakeholders. This allows the team to maintain clear sight on the direction of the project and also reduce the risk of customer unsatisfaction.

The Scrum Way allows for a hands on approach to software development in which the team is prepared for unpredictability and allows for evolution of the original project. This is why it is the most commonly used framework for Agile.

Visually Driven with Kanban

Kanban is Japanese for “visual sign” or “card”, which makes understanding this framework all the more easier. Originally practiced by Toyota line-workers in the manufacturing process to facilitate communication between team members. Now it is used by multiple companies due to its visual workflow.

Teams using Kanban rely more on their eyes to keep track of their progress than most other teams. Kanban takes advantage of the human brains power of visual processing by having teams create a visual representation of their workload. By visual representation I mean a whiteboard full of color and eye candy. Teams respond more effectively to a visual representation in which the work to be done is outlined by category and importance.

Kanban promotes work-in-progress(WIP) principles to help teams analyze future work and estimate how long it would take. This approach also helps reduce the risk of impediments in the project workflow. With this workflow teams focus on one single piece of work rather than a backlog of tasks.

When Kanban is in full motion it is very flexible and allows for changes at any time during the project. Therefore it is the perfect framework if your project is constantly changing tracks and you want to create a more interactive team dynamic.

Gain XP through Extreme Programming

Extreme Programming(XP) is a unique agile framework which strongly emphasizes customer satisfaction and team collaboration more than anything. In Extreme Programming teamwork is crucial, thus the project team is composed of the developers, managers and customers in order to work col laboratively. XP has five distinct improvement which are employed into its workflow: communication, simplicity, feedback, respect, and courage.

Extreme Programmers value communication with their customers and team members the most. The XP teams always tend to maintain their workload as simple as possible in order to avoid risks and impediments. The team receives feedback on their work from day one and throughout the entirety of the project. By maintaining such a strong sense of teamwork XP creates a more satisfying process through which all team members can benefit.

Although XP stands for Extreme Programming and it sounds as if it was complex, it actually follows a simple set of rules. These rules are divided into different categories to which they pertain, those categories are: planning, managing, coding, designing and testing. When practicing the XP approach of Agile it can be a mix of methods and practices at first, but when the rules are used accordingly it begins to form in XP.

XP in many ways resembles Scrum, but in they are different frameworks with their own subset of rules and methods. Many teams prefer the XP route where they have to learn it themselves and adapt it to their project in order to get the full benefits. In a way you could say you gain more experience as a developer and as a team when practicing XP.