Agile Project Management: Best Practices and Methodologies

As defined by Gartner, project management is "the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements."

Project management is an integral part of software engineering processes, along with business analysis, requirement specification, design, programming, and testing. It has been a topic of considerable debate for years. Even today, when project management practices are becoming more mature, only 53 percent of organizations are fully aware of the importance of these practices.

Whatever the industry, project management is a crucial element of a company's efficiency. In fact, organizations using proven project management practices waste 28 less money and implement projects that are 2.5 times more successful.

Project management professionals define a successful project as not only the one that is completed on time and within budget but also as the one that delivers expected benefits.

Project management phases

Regardless of the scope, any project should follow a sequence of actions to be controlled and managed. According to the Project Management Institute (PMI), a typical **project management process** includes the following phases:

1. Initiation
2. Planning
3. Execution
4. Performance monitoring
5. Project close

Used as a roadmap to accomplish specific tasks, these phases define the project management lifecycle.

Yet, this structure is too general. A project usually has a number of internal stages within each phase. They can vary greatly depending on the scope of work, the team, the industry, and the project itself.

In attempts to find a universal approach to managing any project, professionals have developed numerous PM techniques and methodologies.

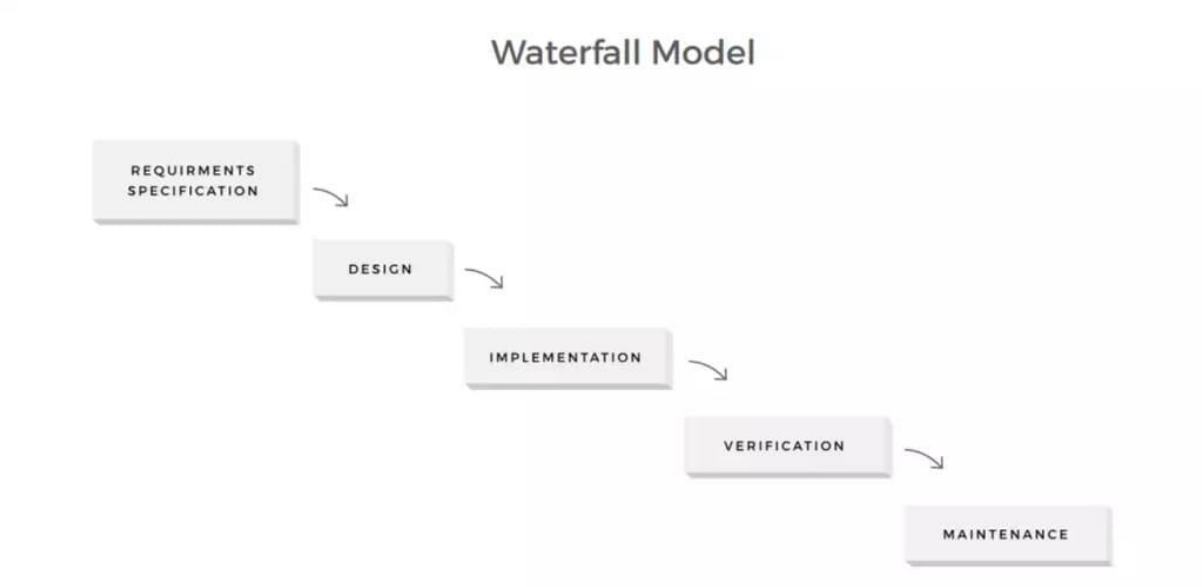
Traditional project management methodology

Based on the above-described classic framework, traditional methodologies take a step-by-step approach to the project execution. Thus, the project goes through the initiation, planning, execution, and monitoring straight to its closure in consecutive stages.

Often called **linear**, this approach includes a number of internal phases that are sequential and executed in chronological order. Applied most commonly within the construction or

manufacturing industry, where little or no changes are required at every stage, traditional project management has found its application in software engineering as well.

Known as the **Waterfall model**, it has been a dominant software development methodology since the early 1970s, when formally described by Winston W. Royce: "*There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed second by a coding step … This sort of very simple implementation concept is in fact, all that is required if the effort is sufficiently small and if the final product is to be operated by those who built it – as is typically done with computer programs for internal use.*"



The Waterfall model has a strong emphasis on planning and specifications development, which takes up to **40 percent of the project time and budget**. Another basic principle of this approach is the strict order of the project phases. A new project stage does not begin until the previous one is finished. The method works well for clearly defined projects with a single deliverable and fixed deadline. The Waterfall approach requires thorough planning, extensive project documentation, and tight control over the development process. In theory, this should lead to on-time, on-budget delivery, low project risks, and predictable final results.
However, when applied to the actual software engineering process, the Waterfall method tends to be slow, costly, and inflexible due to numerous restrictions. In many cases, its inability to adjust the product to the evolving market requirements often results in a huge waste of resources and the eventual project failure.

What is Agile project management

have adopted Agile in one form or another. At the same time, there's a lot of work left to make the practice mature. The history of Agile can be traced back to **1957**: At that time, Bernie Dimsdale, John von Neumann, Herb Jacobs, and Gerald Weinberg were using incremental

development techniques (which are now known as Agile), building software for IBM and Motorola. Not knowing how to classify the approach they were practicing, they realized clearly that it was different from Waterfall in many ways.

The modern-day Agile was officially introduced in **2001** when a group of 17 software development professionals met to discuss alternative project management methodologies. Having a clear vision of the flexible, lightweight, and team-oriented software development approach, they mapped it out in the Manifesto for Agile Software Development.

Aimed at "uncovering better ways of developing software," the Manifesto clearly specifies the Agile fundamentals:
"Through this work, we have come to value: **Individuals and interactions** over processes and tools **Working software** over comprehensive documentation **Customer collaboration** over contract negotiation **Responding to change** over following a plan." Complemented with the Twelve Principles of Agile Software, the philosophy has come to be a universal and efficient new way of managing projects.

Agile approach and process

Agile methodologies take an iterative approach to software development. Unlike a straightforward linear Waterfall model, Agile projects consist of a number of smaller cycles. Each one of them is a project in miniature: it consists of design, implementation, testing, and deployment stages within the pre-defined scope of work.
At the end of each cycle, a potentially shippable product increment is delivered. Thus, with every iteration, new features are added to the product, resulting in gradual project growth. With the features being validated early in the development, the chances of delivering a potentially failed product are significantly lower.

Agile software development steps

The end-to-end Agile project management flow consists of five distinct phases that mostly correspond to the general PM stages we mentioned above.

Envision or initiation phase. The first stage within an Agile project management methodology is about identifying the needs of the end customers, setting business objectives, and outlining the desired results. A project manager identifies the right stakeholders and assigns roles across the team.

Speculation or planning phase. The stage has two main goals: breaking the project into milestones and setting timelines. To achieve the first objective, you need at least a general understanding of project functional requirements. The Agile team prioritizes features and estimates how long it will take to develop them. The phase results in creating an execution plan that, unlike in the Waterfall scenario, will further adapt to changes.

Exploration phase. It involves exploring different ways to address project requirements while staying within time and budget constraints. Once the best option is decided on, the team adds a portion of user stories to an iteration plan and proceeds to their development and testing. The exploration phase goes in parallel with the fourth adaptation phase since the team considers customer feedback and learns from the previous experience.

Adaptation phase. This stage is unique to Agile software development. It enables the team to review the results of previous iterations, assess the existing situation, gather customer feedback, and check performance against the execution plan. Then, you can adapt your plans and approaches accordingly, introducing all needed changes and new requirements if there are any.

Closing phase. At the final stage, the team makes sure that the project is completed and meets all updated requirements. The best practice here is to discuss mistakes occurred during the project and areas for improvements to make better decisions in the future.

Agile best practices

Let's summarize the best practices Agile sticks to.

- **Flexibility:** The scope of work may change according to new requirements.
- **Work breakdown:** The project consists of small cycles.
- **Value of teamwork:** The team members work closely together and have a clear vision of their responsibilities.
- **Iterative improvements:** There is a frequent reassessment of the work done within a cycle to make the final product better.
- **Cooperation with a client:** A customer is closely engaged in the development and can change the requirements or accept the team's suggestions.

According to the 16th Annual State of Agile Report, respondents who have adopted the Agile approach mention the following benefits:

- accelerated time-to-market (52 percent);
- delivery predictability(44 percent); and
- lower risk(31 percent).

Companies also admit that high-performing Agile teams have people-centric values, clear culture tools, and leadership empowerment that positively influence the entire organization.

Agile project management frameworks and methodologies

Agile is an umbrella term for a vast variety of methodologies and techniques, that share the principles and values described above. Each of them has its own areas of use and distinctive features. The most popular frameworks and practices are Scrum, Kanban, Hybrid, Lean, Bimodal, XP, and Crystal. Before discussing them in more detail, let's examine their key features.

Framework	Key Features
Scrum	• The entire scope of work is broken down into short development cycles – Sprints. • The Sprint's duration is from one to four weeks. • The team should strictly follow a work plan for each Sprint. • People involved in a project have predefined roles.
Kanban	• Development is built on workflow visualization. • The current work (work in progress or WIP) is prioritized. • There are n timeboxed development cycles. • The team can change the work plan at any time.
Hybrid	• Agile and Waterfall complement each other. • Agile software development is held under Waterfall conditions (fixed deadline, forecasted budget, and thorough risk assessment).
Bimodal	• There are two separate modes of work – traditional (Mode1) and Agile (Mode 2). • Two separate teams are working on projects with two different goals. • The Mode 1 team maintains IT system infrastructure. • The Mode 2 team delivers innovative applications. • Cross-team collaboration is important.
Lean	• The framework promotes fast software development with less effort, time, and cost. • The development cycle is as short as possible. • The product delivered early is being continuously improved. • The team is independent and has a wider range of responsibilities than those in Scrum, Bimodal, and Hybrid. • Developers can also formulate the product's concept.
XP	• The focus is on the technical aspects of software development. • XP introduces engineering practices aimed at helping developers write clear code. • Product development includes consistent stages: core writing, testing, analyzing, designing, and continuous integration of code. • Face-to-face communication within the team and customer involvement in development are crucial.
Crystal	• The focus is on people and their interactions over processes. • Different teams function differently depending on team size and project priority. • Depending on the number of team members, the framework has several variants including crystal clear, crystal yellow, crystal orange, and crystal red. • Crystal allows for early, frequent shipment of working software while removing bureaucracy and distractions.

Scrum: Most popular framework to Regularly Ship Releases

Scrum is a dominant methodology that dictates the process flow in 87 percent of organizations using Agile. First described in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in the New Product Development Game, it was formulated almost a decade later.

In 1995, Ken Schwaber and Jeff Sutherland, the authors of The Scrum Guide, presented it at the OOPSLA conference. The presentation was based on the knowledge they acquired as they applied the method during the previous few years. While Scrum was introduced far before the Agile Manifesto, it relies on Agile principles and is consistent with the values stated in that document.

Scrum team, roles, and accountabilities

Scrum is aimed at sustaining strong collaboration between people working on complex products and details being changed or added. It is based upon the systematic interactions between the three major roles with defined accountabilities: Scrum Master, Product Owner, and Development Team.

Scrum Master is a central figure within a project. Their principal responsibility is to eliminate all the obstacles that might prevent the team from working efficiently.

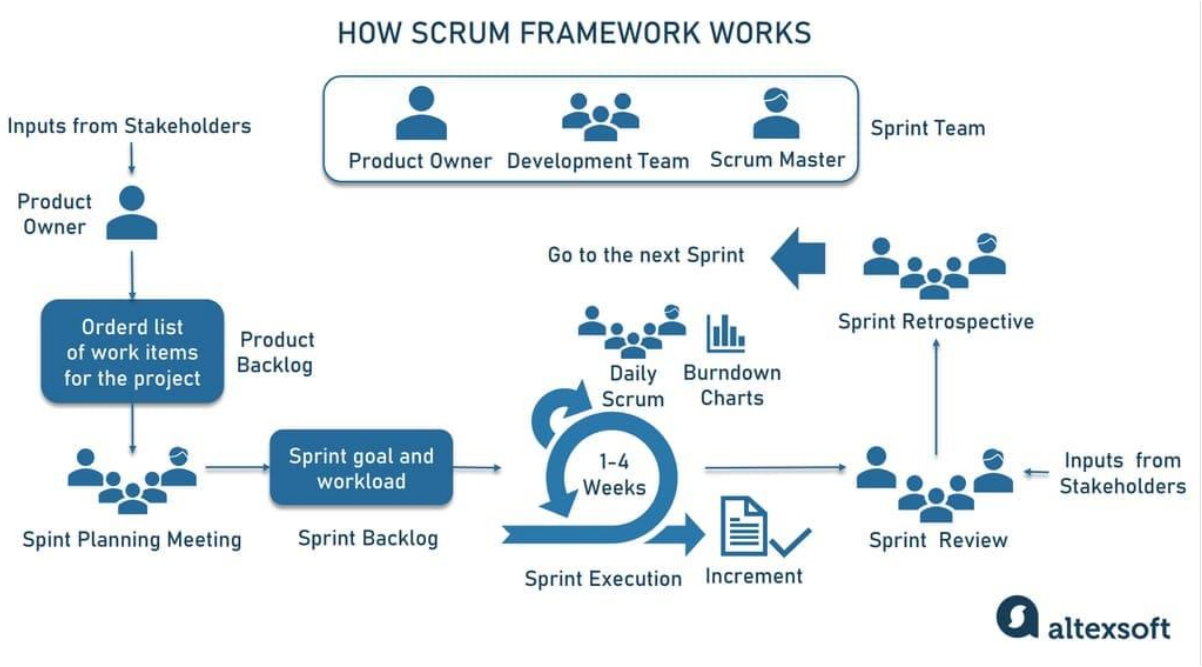
Product Owner, usually a customer or other stakeholder, is actively involved throughout the project, conveying the global vision of the product and providing timely feedback on the job done after every Sprint.

Development Team consists of software engineers, QA engineers, UI/UX designers, and other specialists. They are self-organized, which suggests autonomy in making decisions on how to complete tasks within the Sprint.

Together, these three roles make up the **Scrum Team** — a cross-functional group of people that is responsible for product implementation. It should consist of up to 7 team members to stay flexible and productive.

Sprints and artifacts

A basic unit of work in Scrum – **Sprint** – is a short development cycle that is needed to produce a shippable product increment. A Sprint is usually between 1 and 4 weeks long. More lengthy iterations lack the predictability and flexibility that are scrum's fundamental benefits. Having no standard duration (as long as it is less than 4 weeks), all the Sprints within a project should have a fixed length. This makes it easier to plan and track progress.



Scrum relies on three main artifacts that are used to manage the requirements and track progress – Product Backlog, Sprint Backlog, and Sprint Burndown Chart.

The **Product Backlog** is an ordered list of backlog items that describe the needs of end users or businesses to be met by a final product. The document captures project requirements and goals. It's maintained by the Product Owner, who makes updates each time new information appears. This includes changes in business priorities, new feature requests, feedback from end users, etc.

The **Sprint Backlog** is a list of tasks the team must complete to deliver an increment of functional software at the end of each Sprint. In other words, team members agree on which product items to deliver and define a plan on how to do it.

The **Sprint Burndown Chart** is an illustration of the work remaining in a Sprint. It helps both the Team and the Scrum Master as it shows progress on a day-to-day basis and can predict whether the Sprint goal will be achieved on schedule.

Scrum meetings

The process is formalized through a number of recurring meetings or events, like the Daily Scrum (Standup), the Sprint Planning, the Review, and Retrospective Meetings (the Sprint Retrospective).

The **Daily Scrum** is a timeboxed meeting during which the Scrum Team coordinates its work and sets a plan for the next 24 hours. The event lasts 15 minutes and should be held daily at the same place and time.

The work to be completed is planned at the **Sprint Planning**. Everyone involved in the Sprint (the Product Owner, the Scrum Master, and the Scrum Team) participates in this event. The

Product Owner presents backlog items with the highest priority. The Scrum Team breaks them down into manageable tasks, defines the scope of work for the Sprint, and creates the Sprint Backlog. The Sprint Planning lasts no longer than eight hours for a one-month Sprint. For shorter Sprints, the meeting usually takes less time.

At the end of each Sprint, the Team and the Product Owner meet at the **Sprint Review**. During this informal meeting, the team shows the work completed and answers questions about the product increment. All participants collaborate on what to do next to increase the product's value. The Sprint Review is a four-hour timeboxed meeting for one-month Sprints.

The whole team goes to **Retrospective Meetings** to reflect on their work during the Sprint. Participants discuss what went well or wrong, find ways to improve, and plan how to implement these positive changes. The Sprint Retrospective is held after the Review and before the next Sprint Planning. The event's duration is three hours for one-month Sprints.

When to use Scrum

Scrum works well for long-term, complex projects that require stakeholder feedback, which may greatly affect project requirements. So, when the exact amount of work can't be estimated, and the release date is not fixed, Scrum may be the best choice. The list of companies using this approach is impressive. In fact, there is a public spreadsheet with such organizations, including Microsoft, IBM, Yahoo, and Google.

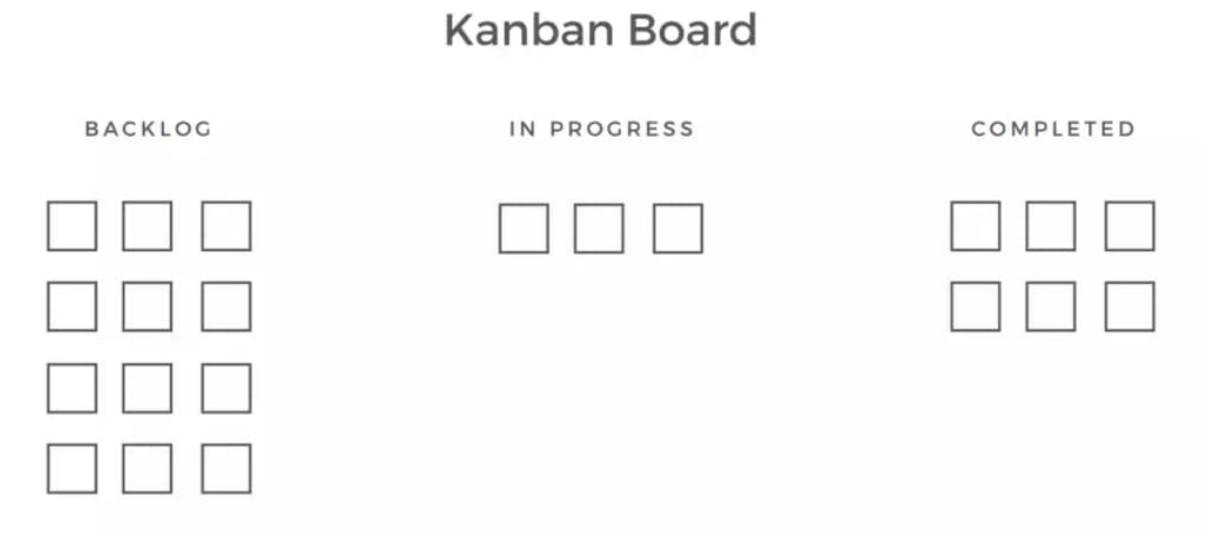
Scrum goes beyond IT. Companies working in the fields of finance, consulting, education, retail, media, and entertainment choose this approach to organize their work processes and enhance cooperation with customers.

Kanban: Visualizing workflows

Kanban is the second most popular framework used by 56 percent of companies practicing Agile. The origins of this simple yet powerful methodology go down to a visual system of cards utilized in Toyota manufacturing as a production control method.

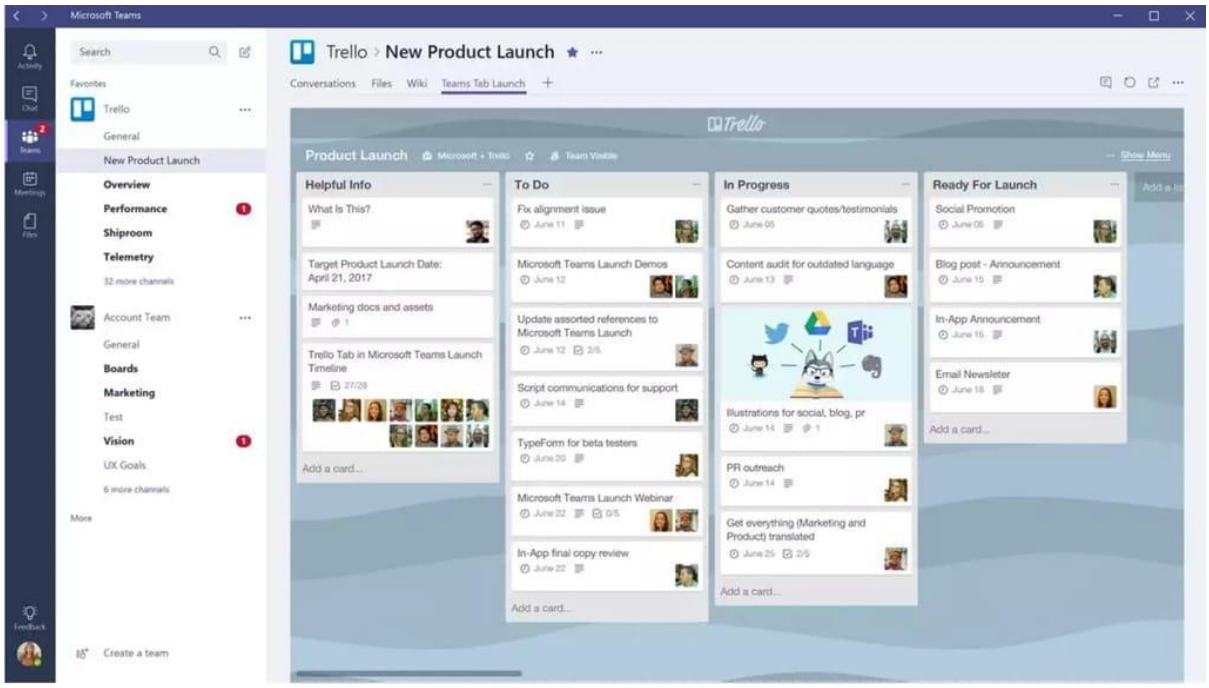
Kanban working principles

Translated as *"visual signal"* from Japanese, Kanban focuses on the visualization of the workflow and prioritizes the **work in progress (WIP)**, limiting its scope to match it effectively to the team's capacity.



As soon as a task is completed, the team can take the next item from the pipeline. Thus, the development process offers more flexibility in planning, faster turnaround, clear objectives, and transparency.

No standard procedures and fixed iterations are required in Kanban, as opposed to Scrum. The project development is based on workflow visualization through a **Kanban board**, usually represented by sticky notes and whiteboards or project management tools like Trello.



Trello automates and digitalizes Kanban. Due to the succinct information about a work item each Kanban card contains, everyone in the team knows who is responsible for the item, what each person's task is, when it's supposed to be finished, etc. Team members can also leave comments and attach screenshots, documents, or links to provide more details.
The ability to track progress helps coworkers understand everyone's personal input in achieving the common goal, resulting in a focus on completing the task well and on time.

When to use Kanban

Using Kanban, teams can do small releases and adapt to changing priorities. Unlike Scrum, there are no Sprints with their predefined goals. Kanban is focused on doing small pieces of work as they come up. For example, if testers find errors in the product, developers try to fix them right away. Kanban, for instance, works well after the main release of the product and suits update and maintenance purposes.

Companies like Spotify and Wooga (a leading mobile games development company) have been using this approach successfully over the years. Yet, 27 percent of organizations combine Scrum with Kanban techniques, using so-called **Scrumban** rather than the original frameworks.

Hybrid: Blending Waterfall and Agile for flexible development and thorough project planning

Agile and Waterfall are two different visions of software development management. The former is about iterative development and being flexible, while the latter, promoting step-by-step development, requires careful planning and rejects making changes along the way. Many companies realize that applying both approaches can be more beneficial than choosing one of the two. The combination of the traditional Waterfall project management and Agile is called Hybrid.

Hybrid framework working principles

Specialists take advantage of the Agile philosophy for software development. But when it comes to budgeting, planning, and hardware design, Waterfall usually works better. So, you can incorporate the most suitable approach depending on your tasks.

To make the most of both methodologies, you may also embed Agile practices into traditional Waterfall work processes. For example, project planning can be done in sprints with feedback gathered regularly. Other ways of modifying the Waterfall model include using Kanban boards and organizing retrospectives.

When to use Hybrid

Hybrid is an effective solution when product development includes both hardware and software. But, there is another reason to choose Hybrid. A situation in which a customer is not satisfied with an unspecified timeframe and budget, as well as a lack of planning, is not rare. Such uncertainty is typical for Agile. In this case, planning, requirements specification, and an application design can be accomplished in Waterfall while Agile is in place for software development and testing.

Bimodal IT: Balancing stability and innovations

The term "Bimodal IT" was introduced by Gartner in 2014.It became one of the popular Agile-Waterfall combinations, allowing companies to handle two IT delivery flows with different methods and goals.

Bimodal working principles

Bimodal IT is the practice of managing two separate but consistent work styles: one focused on predictability and the other on agility.

Mode 1 is traditional; thus, it works perfectly in well-understood and predictable areas. According to Gartner, it focuses on exploiting what is known while transforming the legacy environment into a state fit for a digital world.

Mode 2 involves rapid application development. It is exploratory, nonlinear, and optimized for solving new problems. Mode 2 is especially useful for working on projects that need to be finished as quickly as possible.

Both modes require different skills, techniques, and tools. Therefore, two separate workgroups are needed. These teams have two distinct goals — ensuring stability while adopting innovations. Team members focus on projects that suit their mode best.

The Mode 1 team develops and maintains applications and core systems to support long-term business needs. A company's technological capabilities depend directly on the work that's done by this team.

The Mode 2 team frequently delivers innovative applications to engage new customers and meet short-term business needs. This team may change the product's functionality after having received feedback and analyzed the market.

The teams use different delivery mechanisms and report through different organizational structures. Nevertheless, they need to communicate with each other to exchange ideas and share results.

As Sandy Kemsley specifies, Mode 2 relies on the information and services infrastructure provided by Mode 1, while Mode 1 relies on Mode 2 for testing both new product ideas and new development methods that may eventually be rolled back into Mode 1.

When to use Bimodal

If the company specializes in long- and short-term projects requiring different development and management approaches, Bimodal IT might be the right choice. This framework is about keeping the balance between supporting stable IT infrastructure and driving innovations. Mode 1 ensures a predictable roadmap for legacy system maintenance and infrequent modifications.

At the same time, Mode 2 enables effective experimentation, prototyping, and rapid development.

Lean: Eliminating Waste in Software Engineering

According to the latest estimates, the adoption of **Lean** among companies using Agile increased from just 2 percent in 2015 to 10 percent in 2022.T Having the same origins as Kanban, the approach started as a technique applied to physical manufacturing. It stemmed from the Toyota Production System as a management approach aimed at "*making the vehicles ordered by customers in the quickest and most efficient way, in order to deliver the vehicles as quickly as possible.*"

Lean working principles

The application of Lean principles to software development was initially introduced by Mary and Tom Poppendieck in their book Lean Software Development: An Agile Toolkit.

It includes the **7 basic principles**:

- eliminate waste,
- amplify learning and create knowledge,
- decide as late as possible,
- deliver as fast as possible,
- empower the team,
- build integrity/quality in, and
- see the whole.

Now, let's have a closer look at each of them.

Eliminate waste. In terms of software development, the word "waste" refers to anything that is not adding value to the project and thus should be eliminated. This can be idle time, unnecessary features, or defects.

Amplify learning and create knowledge. In Lean, software development is perceived as an ongoing learning process. Developers don't usually write clear code on the first try. After having detected and fixed errors, they write an improved variation of the previous code. Engineers gain knowledge during development by solving problems and producing code variations. So, the best way to improve the software development environment is to amplify learning.

Decide as late as possible. Late decisions are more informed because they are based on facts. Since technologies become obsolete increasingly faster, delaying an irreversible design decision is a wise move. A major strategy for making commitments late is to reserve the capacity for the change in the system.

Deliver as fast as possible. The fourth principle is about the pros of fast software development. Short development cycles allow developers to learn more by getting feedback. They also allow a customer to delay making a final decision about design until they know more. So, fast delivery helps eliminate waste.

Empower the team. Developers should have the right to make technical decisions as they understand the details of their work like no one else. They can create a roadmap and follow it.

Build integrity/quality in. The user's perception of the software and its characteristics must coincide. If a customer thinks that software has all the needed features and is easy to use, that system has perceived integrity. Conceptual integrity means that the software has a coherent architecture and scores high on usability and fitness of purpose. It can be maintained, adapted, and extended.

See the whole. Engineers should take charge of the overall efficiency of the system instead of focusing on their small portion. If experts adhere to this principle, they can create a system with integrity.

These fundamentals perfectly describe Lean philosophy: Its aim is to deliver more value through less effort, investment, and time.

Lean software development is an **iterative and incremental framework**. Therefore, as in any other Agile approach, the working product increment is delivered at the early stages of development. The further progress depends largely on the product owner's feedback.

What differentiates the Lean approach is that the team is not restricted to any formal processes, such as recurring meetings or thorough task prioritization.

When to use Lean

Lean allows companies to follow a minimum viable product (MVP) development technique. It includes the deployment of a product with a minimum, sufficient set of features to satisfy early users. The idea of the MVP strategy is to gather and analyze customer feedback to know if they like this product and want to buy it. Knowledge of customers' habits, tastes, and needs is the key to producing commercially successful products. Developers use feedback to create a roadmap for future development.

Lean works well for small, short-term projects due to their short life cycles. This approach is also appropriate if the customer can participate in a project realization, as Lean requires ongoing feedback.

Being effectively adopted by a vast number of manufacturing companies, like Nike, Ford, and Intel, Lean principles are widely used in different industries. Startups and successful

companies, e.g., Corbis, PatientKeeper, and Xerox, apply Lean software engineering practices to their processes.

Extreme Programming: Agile practices for writing good code

Extreme Programming (XP) differs from the above-mentioned frameworks by its focus on technical aspects of software development, namely quality code. XP is used by 9 percent of companies.

It requires developers to perform a small number of engineering practices on the highest, almost extreme level possible, hence the name., The framework provides Agile teams with tools to optimize the engineering process and help adapt to changing requirements.

XP Working Principles

XP was introduced in the 1990s. Kent Beck, one of the initial signatories of the Agile Manifesto, invented it while working on a Chrysler Comprehensive Compensation System project. He aimed to find ways of doing sophisticated tasks as expeditiously as possible. In 1999, he documented XP techniques in the book *Extreme Programming Explained: Embrace Change*. The most commonly used **XP practices** are:

- test-driven development (TDD),
- code refactoring,
- continuous integration,
- pair programming, and
- coding standards.

Test-driven development is an advanced engineering technique that uses automated unit tests to propel software design processes. As opposed to the regular development cycle, where the tests are written after the code (or not written at all), TDD has a test-first approach. This means that the unit tests are written prior to the code itself.

According to this approach, the test should fail first when there is no code to accomplish the function. After that, the engineers write the code, focusing on the functionality to make the test pass. As soon as it's done, the source code should be improved to pass all the tests. These three steps are often referred to as the Red Green Refactor cycle.

TDD has proven to provide the following benefits.

1. The tests are used to capture any defects or mistakes in the code, providing constant feedback on the state of every software component. Thus, the quality of the final product is increasingly high.
2. The unit tests can be used as an always up-to-date project documentation, changing as the project evolves.

3. Being deeply involved in product development, the team needs to be able to critically analyze it and foresee the planned outcome in order to test it properly. This keeps the team motivated and engaged, contributing to the product quality.
4. With thorough initial testing, debugging time is minimized.

Code refactoring is a common practice in Agile software development. Basically, it's a process of constant code improvement through simplification and clarification. Solely technical, it does not call for any changes in software behavior.

Extending the source code with each iteration, Agile teams use refactoring as a way to weed out code clutter and duplications. This helps prevent software rot, keeping the code easy to maintain and extend.

Continuous Integration (CI) is another practice Agile teams rely on for managing shared code and software testing. We believe CI is an evolutionary development of Agile principles. Instead of doing short iterations, developers can commit newly written parts of a code several times a day, continuously delivering value to users. By making small and incremental updates multiple times a day, teams can release MVPs more quickly.

To verify the quality of the software — through testing — and automate its deployment, teams use CI/CD tools like CruiseControl, Atlassian Bamboo, TeamCity, or Jenkins.

In addition, CI helps maintain the shared code, eliminating the integration issues. Thus, the product's mainline is robust and clean and can be rapidly deployed.

Pair Programming, or "pairing," is considered to be a very controversial Agile practice. This technique requires two engineers to work together. While one of them is actually writing the code, the other one is actively involved as a watcher, making suggestions and navigating the first through the process.

Being focused on both code and more abstract technical tasks, this team of two is expected to be more efficient, creating better software design and making fewer mistakes. Another benefit of this approach lies in spreading the project knowledge across team members.

However, this practice has often been accused of having a negative impact on the team's short-term productivity. One of the surveys shows that collaborative work usually requires 15 percent more time than individual work, which is a major drawback of the approach. Yet, there are some opinions that the extra time is easily compensated in the long term through the overall higher quality of the software.

Coding standards are one of the most crucial – yet often forgotten – practices in extreme programming. They oblige developers to use standardized formats and styles when writing code. Thus, all team members are capable of reading, sharing, and refactoring code as well as tracking who has worked on certain parts of code.

When to use XP

XP provides tools to decrease risks while designing a new system, especially when developers must write code within strict timeframes. It's essential to know that XP practices are meant for small teams that don't exceed 12 people. One should choose this framework if sure that not only developers but also customers and managers will be able to work together on a project.

XP suggests unit testing as well. If programmers have enough experience creating functional tests, then XP can be used.

Extreme Programming offers engineering practices and ideas that help development teams adapt to ever-changing requirements. The key features of this framework are a high rate of customer engagement and short iterative cycles that don't exceed one week. Also, XP suggests developers make the simplest design possible and prioritize tasks.

While XP can be used as an independent framework, some of its technical practices have become a part of other Agile approaches. Thirteen percent of companies choose the **Scrum/XP Hybrid** framework, where XP engineering practices coexist with Scrum management approaches. For instance, Hybrid includes Scrum events and artifacts. The customer role evolves: It defines a Product Backlog and works together with a Scrum Team in the office until the project ends.

Crystal: Team size matters

Crystal is one of the most adaptive and lightweight approaches to managing software development projects. By its nature, Crystal is a family of Agile methodologies that focuses on people and their interactions over processes, putting such things as skills, communication, and talent first. Developed by IBM employee Alistair Cockburn in 1991, the Crystal methodology suggests that various teams function differently depending on their size and project priority.

Say, members of smaller teams are more likely to be in sync with one another, so they can do without constant reporting and much documentation. On the other hand, larger teams require a more structured communication approach to be on the same page.

There are a few variants of the Crystal framework, each represented by a different color:

- **Crystal clear** – teams with up to 8 people,
- **Crystal yellow** – teams ranging from 10 to 20 people,
- **Crystal orange** – teams ranging from 20 to 50 people, and
- **Crystal red** – big teams ranging from 50 to 100 people.

Crystal Working Principles

Similar to the above-mentioned Agile practices and frameworks, Crystal allows for early, frequent shipment of working software while removing bureaucracy and distractions.

There are **key 7 principles of the Crystal Agile method**.

Frequent delivery. Teams must regularly deliver working software functionalities to users, ensuring a real-time view of whether a product meets the customer's needs.

Reflective improvement. Teams take time to reflect on the work they already did to detect what should be improved and why.

Osmotic communication. Teams must be located in the same physical space to enable seamless information flow between all members, like molecule movement in osmosis.

Personal safety. All members of a team should be vocal about their opinions without fear of being questioned or ridiculed. There should be a high level of trust within a team.

Focus on work. Leaders should decide on priorities and provide developers with dedicated time, during which they can focus on their work with no interruptions, meetings, or demos.

Access to subject matter experts and users. Teams must get direct access to feedback from real users when they need it.

Access to the technical environment. Development teams should have the required tooling for continuous deployment and automated testing to fix bugs and errors.

Crystal Agile framework facilitates team communication while allowing for adaptability to the demanding requirements. At the same time, there may be a loss of focus due to the lack of pre-defined plans and structure.

When to use Crystal

Crystal is the right approach for experienced development teams that lack autonomy in decision-making, as it allows building processes in a way that works best for each team. For teams that work remotely, it’s better to choose another framework because Crystal requires direct communication.

Agile methodology implementation steps

Once you have decided that the Agile methodology fills the bill in terms of your company and projects, you need to know how to implement it successfully. While the process may differ from one company to another, there are a few general steps to follow.

Step 1: Get your manager and stakeholder buy-in

Before you start implementing the new way of doing projects, make sure that everyone is on the same page and supports the change. So, it's a good idea to talk to the key players and get their buy-in by explaining the benefits of Agile, addressing any of their concerns, and answering questions.

Step 2: Start small

Since incremental progress is the cornerstone of Agile, it makes sense to begin the methodology implementation with one small project, evaluate the feedback, and then apply it to other projects within your organization.

Step 3: Get your team excited

The success of Agile projects relies on the ability of different team members to cooperate and communicate. If your team isn't excited about the whole idea and/or doesn't support change, implementing Agile will be a tough task. After all, one of the main principles of Agile puts individuals and their interactions over processes and tools.

Step 4: Choose a fitting framework and stick to it

As you see, there are a lot of different Agile frameworks and practices to use. Each one has different requirements and focus. It's important to pick an Agile framework that fits your processes in the best way and stick to it. If, say, you decide to implement Scrum, make sure your team strictly follows a work plan for each Sprint and attends daily meetings.

Conclusion

The Agile approach is often mistakenly considered to be a single methodology. Yet, there are dozens of methodologies and certain practices that have not been touched upon in this research.

Regardless of the exact methodologies and techniques they use, Agile teams have proven to increase profits 37 percent faster and generate 30 percent more revenue than non-Agile companies. Higher speed, flexibility, and productivity achieved through such approaches are the key drivers that motivate more and more organizations to switch to Agile.

Software engineering, being an extremely fast-paced industry, calls for flexibility and responsiveness in every aspect of project development. Agile methodologies allow for delivering cutting-edge products and cultivating innovative experiences while keeping the product in sync with market trends and user requirements.

However, there is always a place for diversity. Depending on your business requirements and goals, you might still benefit from using the Waterfall model or the combination of the two.

References

1. Pulse of the Profession 2017 – http://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf?sc_lang_temp=en
2. Gartner glossary: Project Management – https://blogs.gartner.com/it-glossary/project-management/
3. Managing the Development of Large Software Systems – http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf
4. 16th Annual State of Agile Report – https://info.digital.ai/rs/981-LQX-968/images/SOA16.pdf
5. The new product development game – https://hbr.org/1986/01/the-new-new-product-development-game
6. Iterative and Incremental Development: A Brief History – https://www.cs.umd.edu/~basili/publications/journals/J90.pdf
7. Manifesto for Agile Software Development – http://www.agilemanifesto.org/
8. The Scrum Guide – http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100
9. Object-oriented Programming, Systems, Languages and Applications – http://www.sigplan.org/Conferences/OOPSLA/
10. The State of Scrum Report 2017 - https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/State%20of%20Scrum/State0fScrum_2016_FINAL.pdf
11. Toyota Production System – http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/
12. Gartner glossary: Bimodal – https://www.gartner.com/it-glossary/bimodal/
13. Bimodal IT –https://infotech.report/Resources/Whitepapers/119d385e-41cf-44f3-b518-9b56b277311b_SAG_Bimodal_IT_8PG_WP_Aug16-Web_tcm16-143391.pdf
14. Poppendieck – http://www.poppendieck.com/
15. Lean Software Development: An Agile Toolkit – http://ptgmedia.pearsoncmg.com/images/9780321150783/samplepages/0321150783.pdf
16. Guidelines for Test-driven Development – https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx
17. Strengthening the Case for Pair-Programming – http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware.PDF
18. Programming in Pairs: How to Get Started – https://www.ibm.com/garage/method/practices/code/remote-pair-programming/
19. Extreme Programming: A Gentle Introduction – http://www.extremeprogramming.org/index.html