What's the Difference? Agile vs Scrum vs Waterfall vs Kanban

Sticky Bar Title

Manage Any Project Your Way with Smartsheet
Try Smartsheet for Free (https://www.smartsheet.com/try-it?trp=8586&lx=mhl4L9lVbpuSayzuzPhz7g&lpv=stkcta)
Try Smartsheet for Free (https://www.smartsheet.com/try-it?trp=8586&lx=mhl4L9lVbpuSayzuzPhz7g&lpv=stkcta)
Over the course of a project, you'll make hundreds of decisions. And one of the first decisions you'll make is choosing which project management methodology to follow.

From Agile to Scrum to Waterfall to Kanban, there are a variety of different project management frameworks. Some, like Scrum, follow a more rigid, structured methodology. Others, like Kanban, are easier to introduce and implement on top of existing processes. They all have pros and cons, so how do you know which one to choose?

This article will cover the differences between Agile vs Scrum vs Waterfall vs Kanban. We'll talk about the advantages, disadvantages, stages, and when you should use each one.

Agile Methodology

What Is Agile?

Agile software development is based on an incremental, iterative approach. Instead of in-depth planning at the beginning of the project, Agile methodologies are open to changing requirements over time and encourages constant feedback from the end users. Cross-functional teams work on iterations of a product over a period of time, and this work is organized into a backlog that is prioritized based on business or customer value. The goal of each iteration is to produce a working product.

In Agile methodologies, leadership encourages teamwork, accountability, and face-to-face communication. Business stakeholders and developers must work together to align the product with customer needs and company goals.

Agile refers to any process that aligns with the concepts of the Agile Manifesto. In February 2001, 17 software developers met in Utah to discuss lightweight development methods. They published the Manifesto for Agile Software Development (http://agilemanifesto.org/), which covered how they found “better ways of developing software by doing it and helping others do it” and included four values and 12 principles. The Agile Manifesto is a dramatic contrast to the traditional text A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (http://www.pmi.org/PMBOK-Guide-and-Standards.aspx) and standards.

12 Principles of Agile Methodology

The Agile Manifesto lists 12 principles (http://agilemanifesto.org/principles.html) to guide teams on how to execute with agility. These are the principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the shorter timescale.
Business people and developers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Working software is the primary measure of progress.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Continuous attention to technical excellence and good design enhances agility.
Simplicity -- the art of maximizing the amount of work not done -- is essential.
The best architectures, requirements, and designs emerge from self-organizing teams.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Advantages of Agile

Agile evolved from different lightweight software approaches in the 1990s and is a response to some project managers’ dislike of the rigid, linear Waterfall methodology. It focuses on flexibility, continuous improvement, and speed.

Here are some of the top advantages of Agile:

Change is embraced: With shorter planning cycles, it's easy to accommodate and accept changes at any time during the project. There is always an opportunity to refine and reprioritize the backlog, letting teams introduce changes to the project in a matter of weeks.

End-goal can be unknown: Agile is very beneficial for projects where the end-goal is not clearly defined. As the project progresses, the goals will come to light and development can easily adapt to these evolving requirements.

Faster, high-quality delivery: Breaking down the project into iterations (manageable units) allows the team to focus on high-quality development, testing, and collaboration. Conducting testing during each iteration means that bugs get identified and solved more quickly. And this high-quality software can be delivered faster with consistent, successive iterations.

Strong team interaction: Agile highlights the importance of frequent communication and face-to-face interactions. Teams work together and people are able to take responsibility and own parts of the projects.

Customers are heard: Customers have many opportunities to see the work being delivered, share their input, and have a real impact on the end product. They can gain a sense of ownership by working so closely with the project team.

Continuous improvement: Agile projects encourage feedback from users and team members throughout the whole project, so lessons learned are used to improve future iterations.

Disadvantages of Agile

While the level of flexibility in Agile is usually a positive, it also comes with some trade-offs. It can be hard to establish a solid delivery date, documentation can be neglected, or the final product can be very different than originally intended.

Here are some of the disadvantages of Agile:

Planning can be less concrete: It can sometimes be hard to pin down a solid delivery date. Because Agile is based on time-boxed delivery and project managers are often reprioritizing tasks, it's possible that some items originally scheduled for delivery may not be complete in time. And, additional sprints may be added at any time in the project, adding to the overall timeline.

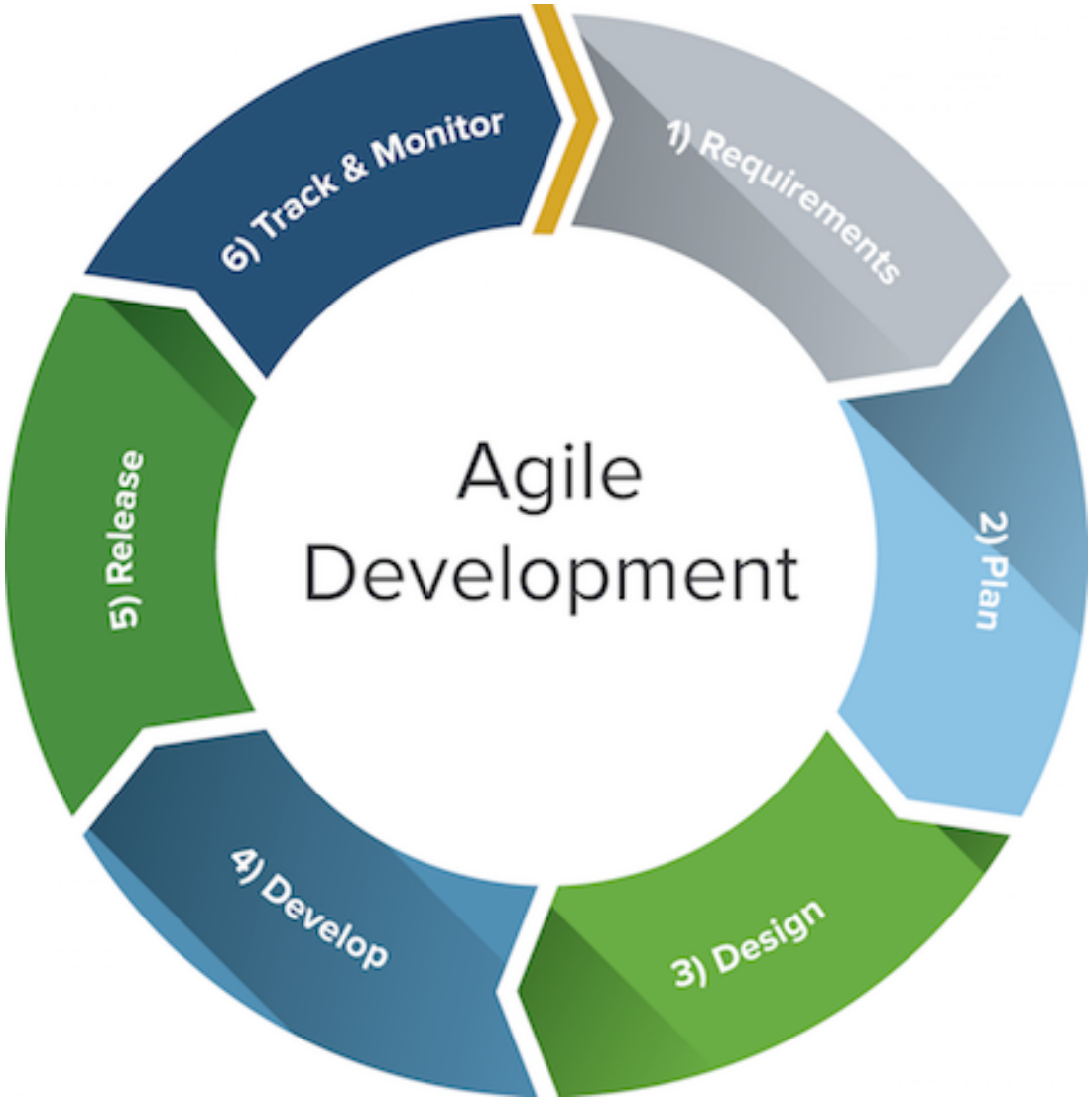
Team must be knowledgeable: Agile teams are usually small, so team members must be highly skilled in a variety of areas. They also must understand and feel comfortable with the chosen Agile methodology.

Time commitment from developers: Agile is most successful when the development team is completely dedicated to the project. Active involvement and collaboration is required throughout the Agile process, which is more time consuming than a traditional approach. It also means that the developers need to commit to the entire duration of the project.

Documentation can be neglected: The Agile Manifesto prefers working software over comprehensive documentation, so some team members may feel like it's less important to focus on documentation. While comprehensive documentation on its own does not lead to project success, Agile teams should find the right balance between documentation and discussion.

Final product can be very different: The initial Agile project might not have a definitive plan, so the final product can look much different than what was initially intended. Because Agile is so flexible, new iterations may be added based on evolving customer feedback, which can lead to a very different final deliverable.

The Agile Development Cycle



Here are the phases in the Agile development cycle. It's important to note that these phases shouldn't happen in succession; they are flexible and always evolving. Many of these phases happen in parallel.

Planning: Once an idea is deemed viable and feasible, the project team comes together and works to identify features. The goal of this phase is to break down the idea into smaller pieces of work (the features) then to prioritize each feature and assign it to an iteration.

Requirements analysis: This phase involves many meetings with managers, stakeholders, and users to identify business requirements. The team needs to gather information like who will use the product and how they will use it. These requirements must be quantifiable, relevant, and detailed.

Design: The system and software design is prepared from the requirements identified in the previous phase. The team needs to think about what the product or solution will look like. The test team also comes up with a test strategy or plan to proceed.

Implementation, coding or development: This phase is all about creating and testing features, and scheduling iterations for deployment (following the iterative and incremental development approach [IID]). The development phase starts with iteration 0, because there are no features being delivered. This iteration lays down the foundation for development, with tasks like finalizing contracts, preparing the environments, and funding.

Testing: Once the code has been developed, it is tested against the requirements to make sure the product is actually solving customer needs and matching user stories. During this phase, unit testing, integration testing, system testing, and acceptance testing are done.

Deployment: After testing, the product is delivered to customers for them to use. However, deployment isn't the end of the project. Once customers start using the product, they may run into new problems that the project team will need to address.

Methodologies That Are Used to Implement Agile

Agile is a framework and there are a number of specific methods within the Agile movement. You can think of these as different flavors of Agile:

Extreme Programming (XP): Also known as XP, Extreme Programming is a type of software development intended to improve quality and responsiveness to evolving customer requirements. The principles of XP include feedback, assuming simplicity, and embracing change.

Feature-driven development (FDD): This iterative and incremental software development process blends industry best practices into one approach. There are five basic activities in FDD: develop overall model, build feature list, plan by feature, design by feature, and build by feature.

Adaptive system development (ASD): Adaptive system development represents the idea that projects should always be in a state of continuous adaptation. ASD has a cycle of three repeating series: speculate, collaborate, and learn.

Dynamic Systems Development Method (DSDM): This Agile project delivery framework is used for developing software and non-IT solutions. It addresses the common failures of IT projects, like going over budget, missing deadlines, and lack of user involvement. The eight principles of DSDM are: focus on the business need, deliver on time, collaborate, never compromise quality, build incrementally from firm foundations, develop iteratively, communicate continuously and clearly, and demonstrate control.

Lean Software Development (LSD): Lean Software Development takes Lean manufacturing and Lean IT principles and applies them to software development. It can be characterized by seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.

Kanban: Kanban, meaning "visual sign" or "card" in Japanese, is a visual framework to implement Agile. It promotes small, continuous changes to your current system. Its principles include: visualize the workflow, limit work in progress, manage and enhance the flow, make policies explicit, and continuously improve.

Crystal Clear: Crystal Clear is part of the Crystal family of methodologies. It can be used with teams of six to eight developers and it focuses on the people, not processes or artifacts. Crystal Clear requires the following: frequent delivery of usable code to users, reflective improvement, and osmotic communication preferably by being co-located.

Scrum: Scrum is one of the most popular ways to implement Agile. It is an iterative software model that follows a set of roles, responsibilities, and meetings that never change. Sprints, usually lasting one to two weeks, allow the team to deliver software on a regular basis.

Other Practices in Agile

There are many other practices and frameworks that are related to Agile. They include:

Agile Modeling (AM): Agile modeling is used to model and document software systems and is a supplement to other Agile methodologies like Scrum, Extreme Programming (XP), and Rational Unified Process (RUP). AM is not a complete software process on its own. It can help improve models with code, but it doesn't include programming activities.

Rational Unified Process (RUP): Created by the Rational Software Corporation, a division of IBM, RUP is an iterative, adaptive framework for software development. According to Rational, RUP is like an online mentor that provides guidelines, templates, and examples for program development. The key aspects of RUP include a risk-driven process, use case focused development, and architecture-centric design.

Lean vs Agile: Lean development focuses on eliminating and reducing waste (activities that don't add any value). Lean development takes the principles from Lean manufacturing and applies them to software development. These principles are very similar to Agile, however Lean takes it one step further. In the development phase, you select, plan, develop, test, and deploy only one feature before you repeat the process for the next feature.

Test-Driven Development (TDD): Test-driven development relies on repetitive, short development cycles. First, a developer writes an (initially failing) automated test case for a new feature and quickly adds a test with the minimum amount of code to pass that test. Then, he refactors the new code to acceptable standards.

Scaled Agile Framework (SAFe trademark logo): The Scaled Agile Framework is a very structured method to help large businesses get started with adopting Agile. SAFe is based on Lean and Agile principles and tackles tough issues in big organizations, like architecture, integration, funding, and roles at scale. SAFe has three levels: team, program, and portfolio.

Rapid Application Development (RAD): RAD's approach to software development puts more emphasis on development than planning tasks. It follows an incremental model, where each component is developed in parallel. The phases in RAD are: business modeling, data modeling, process modeling, application generation, and testing and turnover.

Empirical Control Method: With Agile software development, you can use an Empirical Control Method, which means that you make decisions based on the realities you observe in the actual project. The empirical model of process control has three parts: visibility, inspection, and adaption.

How to Estimate Budgets in Agile

Without in-depth, upfront planning, many project managers are unsure of how to calculate the cost and budget of an Agile project.

Estimating the cost before the project even starts can always be challenging, regardless of which project methodology you use. However, in an Agile project, you can tie the amount of time the project will take with its total cost.

First, create a burndown chart and use the burndown rate to estimate how many sprints will be in your project and when the project will end. Then, calculate how much the team will cost based on their hourly rates. Multiply each person's rate by the number of working hours per week, then multiply that by the number of weeks in a sprint. Once you estimate the initial budget for your team, you can add any other costs, like technology, travel, or equipment.

You could also break down each user story into tasks. Once you have an idea of how many hours it will take to complete each task, you can estimate the project budget.

And lastly, you could use planning poker to estimate the effort required for development goals. Planning poker is a consensus-based, gamified technique for estimating the effort of development goals. Each team member makes estimates by playing numbered cards face-down on the table, instead of saying it out loud. The cards are then revealed and the estimates discussed with the whole team.

Agile and Pair Programming

Pair programming (also known as “pairing”) is part of the Extreme Programming (XP) practices. It is when two programmers share a single workstation, which includes sharing one screen, keyboard, and mouse. The purpose of this technique is to encourage better communication, clarification of the problem, and understanding of the solution. Pairing is often used in Agile projects to quickly deliver high-quality products, but is it always required?

The answer depends on your programmers, company, and goals. For some projects and programmers, pairing might improve productivity. However, it may not always be appropriate for every project. The best thing to do is experiment and see if it works for you.

How Agile Addresses Software Requirements

Agile helps development teams focus on customers’ most important requirements as quickly as possible. With continuous feedback and frequent face-to-face interactions, the project team and stakeholders understand and prioritize the right requirements.

Agile teams use backlogs with user stories to manage requirements. Before an iteration begins, the team agrees on which requirements they should meet with the next delivery. This collaborative approach ensures that the most important features get prioritized. And, requirements are continuously updated throughout the project as new information is surfaced.

Can You Use Agile for Projects Outside of Software?

While Agile was traditionally created for software development, it can also be used in many other projects and industries.

It’s important to remember that Agile software development was born from the principles of Lean manufacturing and organizational learning. These ideas weren’t based on software to begin with. And, many practices in Agile, like stand-up meetings and visual management, are so common and can apply to any industry.

There are not many case studies of teams using Agile for things outside of software, but there are a couple. For example, Kate Sullivan, a corporate lawyer on The Lonely Planet legal team, has transformed the legal affairs service delivery with Agile. The team uses whiteboards and cards, morning stand-up meetings, prioritization, weekly iterations, and regular retrospectives.

Agile can definitely be applied to projects outside of software development, you just have to find the right method and approach for your needs. You can start with boards and cards, a work backlog, stand-up meetings, or iterations (weekly planning meetings) to see how your team responds.

How to Get Started with Agile

A simple way to get started with Agile is to incorporate daily stand-up meetings into your project. Daily stand-up meetings are easy to incorporate into any other project methodology you may already be using (even Waterfall) and don’t require any training or knowledge transfer. Meet at the same spot every day for about ten minutes and have everyone talk about what they worked on the day before, what they’ll work on today, and any roadblocks.

If you want to make the complete switch to Agile all at once, you may want to start with understanding why the team and organization want to make this change. What is and isn’t working? What are they looking to improve? Then, you could conduct an Agile assessment, getting a complete view of the people, skills, and technologies used.

Whichever route you choose, remember that Agile is flexible in its very nature. There is no wrong or right way to get started with Agile. Do what works for you and your team.

Smartsheet's newest view, Card View, gives Agile teams a more highly-visual way to work, communicate, and collaborate in Smartsheet. Card View enables you to focus attention with rich cards, give perspective with flexible views, and prioritize and adjust work more visually. Display information on cards including custom fields, images, and color coding to better focus your team's attention. Categorize cards into lanes to organize your work more visually.

Scrum Methodology

What Is Scrum?

Scrum is a subset of Agile and one of the most popular process frameworks for implementing Agile. It is an iterative software development model used to manage complex software and product development. Fixed-length iterations, called sprints lasting one to two weeks long, allow the team to ship software on a regular cadence. At the end of each sprint, stakeholders and team members meet to plan next steps.

Scrum follows a set of roles, responsibilities, and meetings that never change. For example, Scrum calls for four ceremonies that provide structure to each sprint: sprint planning, daily stand-up, sprint demo, and sprint retrospective. During each sprint, the team will use visual artifacts like task boards or burndown charts to show progress and receive incremental feedback.

Jeff Sutherland created the Scrum process in 1993, taking the term "Scrum" from an analogy in a 1986 study by Takeuchi and Nonaka published in the Harvard Business Review. In the study, Takeuchi and Nonaka compare high-performing, cross-functional teams to the Scrum formation used by Rugby teams. The original context for this was manufacturing, but Sutherland, along with John Scumniotales and Jeff McKenna, adapted the model for software development.

Advantages of Scrum

Scrum is a highly prescriptive framework with specific roles and ceremonies. While it can be a lot to learn, these rules have a lot of advantages. The benefits of Scrum include:

More transparency and project visibility: With daily stand-up meetings, the whole team knows who is doing what, eliminating many misunderstandings and confusion. Issues are identified in advance, allowing the team to resolve them before they get out of hand.

Increased team accountability: There is no project manager telling the Scrum Team what to do and when. Instead, the team collectively decides what work they can complete in each sprint. They all work together and help each other, improving collaboration and empowering each team member to be independent.

Easy to accommodate changes: With short sprints and constant feedback, it's easier to cope with and accommodate changes. For example, if the team discovers a new user story during one sprint, they can easily add that feature to the next sprint during the backlog refinement meeting.

Increased cost savings: Constant communication ensures the team is aware of all issues and changes as soon as they arise, helping to lower expenses and increase quality. By coding and testing features in smaller chunks, there is continuous feedback and mistakes can be corrected early on, before they get too expensive to fix.

Disadvantages of Scrum

While Scrum offers some concrete benefits, it also has some downsides. Scrum requires a high level of experience and commitment from the team and projects can be at risk of scope creep.

Here are the disadvantages of Scrum:

Risk of scope creep: Some Scrum projects can experience scope creep due to a lack of specific end date. With no completion date, stakeholders may be tempted to keep requesting additional functionality.

Team requires experience and commitment: With defined roles and responsibilities, the team needs to be familiar with Scrum principles to succeed. Because there are no defined roles in the Scrum Team (everyone does everything), it requires team members with technical experience. The team also needs to commit to the daily Scrum meetings and to stay on the team for the duration of the project.

The wrong Scrum Master can ruin everything: The Scrum Master is very different from a project manager. The Scrum Master does not have authority over the team; he or she needs to trust the team they are managing and never tell them what to do. If the Scrum Master tries to control the team, the project will fail.

Poorly defined tasks can lead to inaccuracies: Project costs and timelines won't be accurate if tasks are not well defined. If the initial goals are unclear, planning becomes difficult and sprints can take more time than originally estimated.

Roles in Scrum

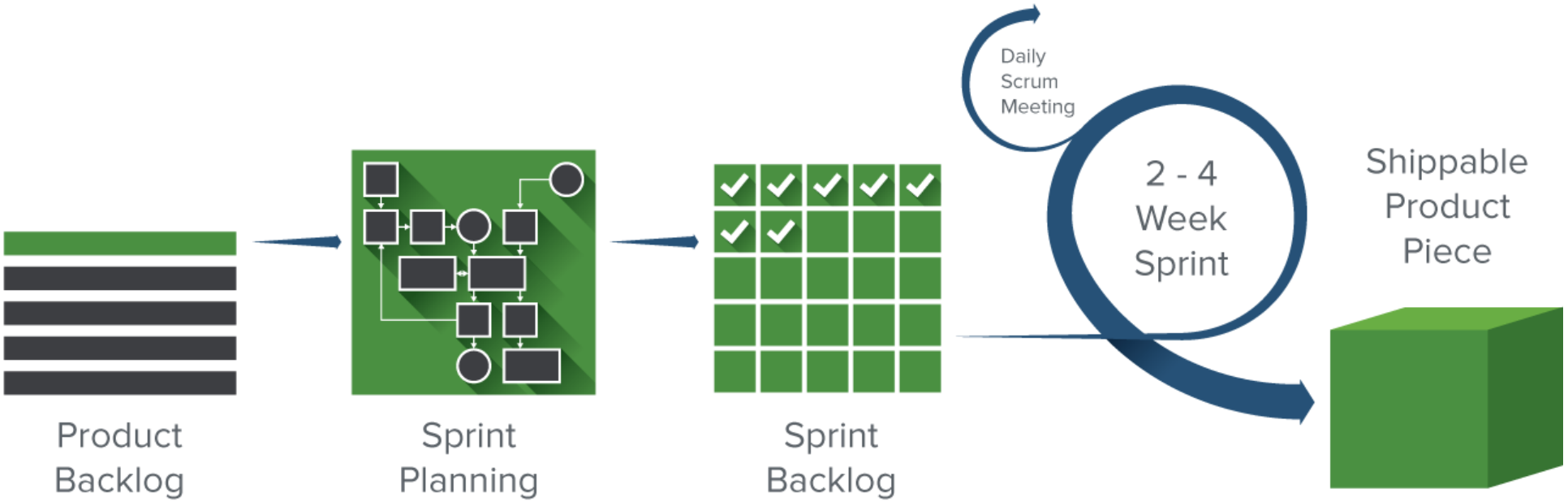
There are three specific roles in Scrum. They are:

Product Owner: The Scrum Product Owner has the vision of what he or she wants to build and conveys that vision to the team. The Product Owner focuses on business and market requirements, prioritizing all the work that needs to be done. He or she builds and manages the backlog, provides guidance on which features to ship next, and interacts with the team and other stakeholders to make sure everyone understands the items in the product backlog. The Product Owner is not a project manager. Instead of managing the status and progress, his or her job is to motivate the team with a goal and vision.

Scrum Master: Often considered the coach for the team, the Scrum Master helps the team do their best possible work. This means organizing meetings, dealing with roadblocks and challenges, and working with the Product Owner to ensure the product backlog is ready for the next sprint. The Scrum Master also makes sure the team follows the Scrum process. He or she doesn't have authority over the team members, but he or she does have authority over the process. For example, the Scrum Master can't tell someone what to do, but could propose a new sprint cadence.

Scrum Team: The Scrum Team is comprised of five to seven members. Everyone on the project works together, helps each other, and shares a deep sense of camaraderie. Unlike traditional development teams, there are not distinct roles like programmer, designer, or tester. Everyone completes the set of work together. The Scrum Team owns the plan for each sprint; they anticipate how much work they can complete in each iteration.

Steps in the Scrum Process



There are a specific, unchanging set of steps in the Scrum flow. They include:

Product backlog: The Product Owner and Scrum Team meet to prioritize the items on the product backlog (the work on the product backlog comes from user stories and requirements). The product backlog is not a list of things to be completed, but rather it is a list of all the desired features for the product. The development team then pulls work from the product backlog to complete during each sprint.

Sprint planning: Before each sprint, the Product Owner presents the top items on the backlog to the team in a sprint planning meeting. The team then chooses which work they can complete during the sprint and moves the work from the product backlog to the sprint backlog (which is a list of tasks to complete in the sprint).

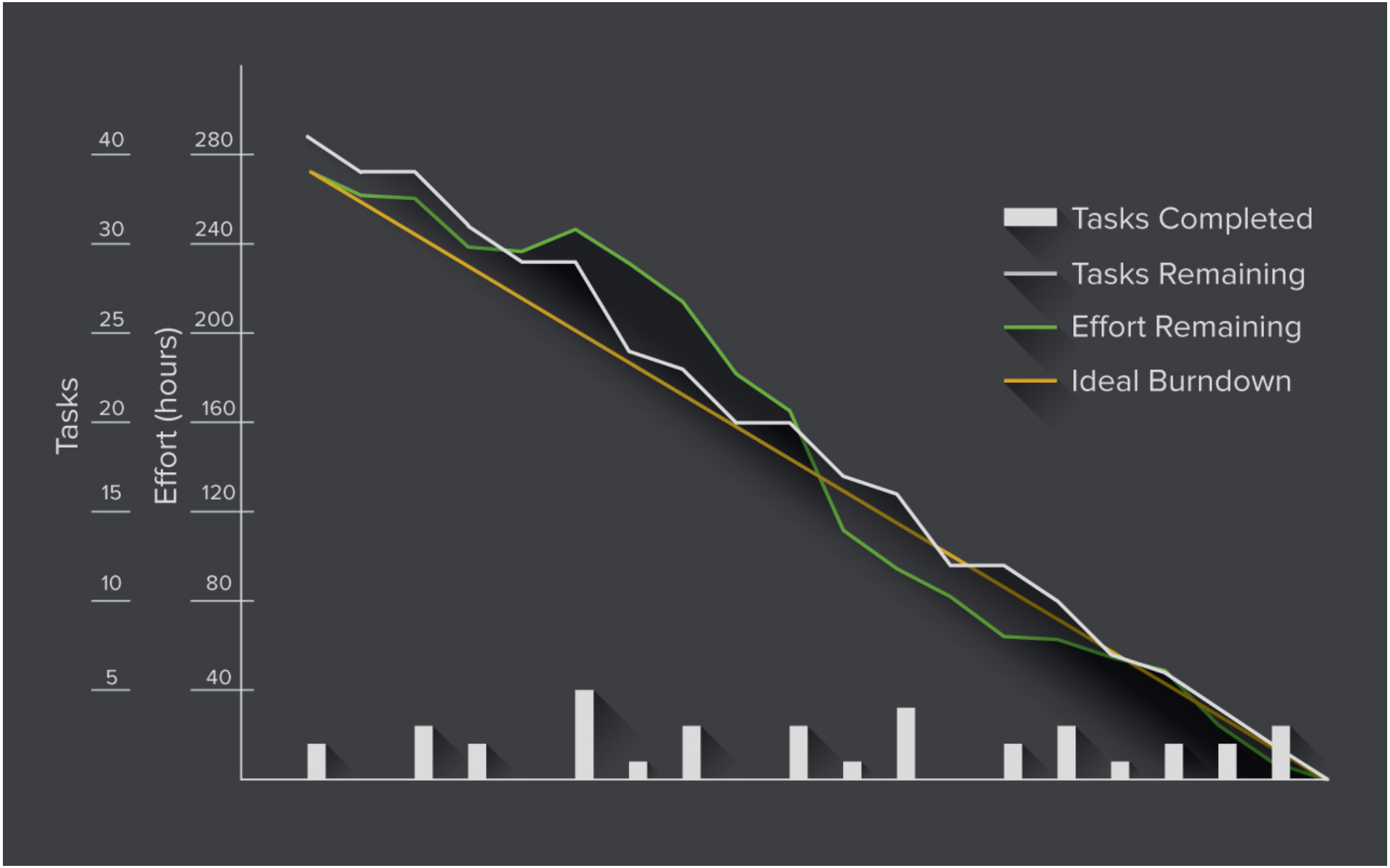
Backlog refinement/grooming: At the end of one sprint, the team and Product Owner meet to make sure the backlog is ready for the next sprint. The team may remove user stories that aren't relevant, create new stories, reassess the priority of stories, or split user stories into smaller tasks. The purpose of this "grooming" meeting is to ensure the backlog only contains items that are relevant and detailed, and that meet the project's objectives.

Daily Scrum meetings: The Daily Scrum is a 15-minute stand-up meeting where each team member talks about their goals and any issues that have come up. The Daily Scrum happens every day during the sprint and helps keep the team on track.

Sprint review meeting: At the end of each sprint, the team presents the work they have completed at a sprint review meeting. This meeting should feature a live demonstration, not a report or a PowerPoint presentation.

Sprint retrospective meeting: Also at the end of each sprint, the team reflects on how well Scrum is working for them and talks about any changes that need to be made in the next sprint. The team may talk about what went well during the sprint, what went wrong, and what they could do differently.

Tools, Artifacts, and Methods in Scrum



In addition to roles and ceremonies, Scrum projects also include certain tools and artifacts. For example, the team uses a Scrum board to visualize the backlog or a burndown chart to show outstanding work. The most common artifacts and methods are:

Scrum board: You can visualize your sprint backlog with a Scrum task board. The board can have different forms; it traditionally involves index cards, Post-It notes, or a whiteboard. The Scrum board is usually divided into three categories: to do, work in progress, and done. The Scrum Team needs to update the board throughout the entire sprint. For example, if someone comes up with a new task, she would write a new card and put it in the appropriate column.

User stories: A user story describes a software feature from the customer's perspective. It includes the type of user, what they want, and why they want it. These short stories follow a similar structure: as a <type of user>, I want to <perform some task> so that I can <achieve some goal.> The development team uses these stories to create code that will meet the requirements of the stories.

Burndown chart: A burndown chart represents all outstanding work. The backlog is usually on the vertical axis, with time along the horizontal axis. The work remaining can be represented by story points, ideal days, team days, or other metrics. A burndown chart can warn the team if things aren't going according to plan and helps to show the impact of decisions.

Large-Scale Scrum (LeSS): If you want to scale elements of Scrum to hundreds of developers, the Large-Scale Scrum (LeSS) framework helps extend the rules and guidelines without losing the core of Scrum. The principles are taken directly from Scrum, however focuses on scaling up without adding additional overhead (like adding more roles, artifacts, or processes).

Timeboxing: A timebox is a set period of time during which a team works towards completing a goal. Instead of letting a team work until the goal is reached, the timebox approach stops work when the time limit is reached. Time-boxed iterations are often used in Scrum and Extreme Programming.

Icebox: Any user stories that are recorded but not moved to development are stored in the icebox.
The term "icebox" was created by Pivotal Tracker, an Agile project management tool.

Scrum vs RUP: While both Scrum and Rational Unified Process (RUP) follow the Agile framework, RUP involves more formal definition of scope, major milestones, and specific dates (Scrum uses a project backlog instead of scope). In addition, RUP involves four major phases of the project lifecycle (inception, elaboration, construction, and transition), whereas Scrum dictates that the whole "traditional lifecycle" fits into one iteration.

Lean vs Scrum: Scrum is a software development framework, while Lean helps optimize that process. Scrum's primary goal is on the people, while Lean focuses on the process. They are both considered Agile techniques, however Lean introduces two major concepts: eliminating waste and improving flow.

How to Get Started with Scrum

Working with Scrum often means changing the team's habits. They need to take more responsibility, increase the quality of the code, and boost speed of delivery. This level of commitment acts as a change agent; as the teams commit to sprint goals, they are more and more motivated to get better and faster to deliver a quality product.

A good place to start with Scrum is to talk about the roles. Every project must have a Scrum Master, Product Owner, and Scrum Team. You may want to talk about who should be the Scrum Master and Product Owner, or if these roles are already assigned, you may want to clarify their roles and responsibilities.

Depending on how familiar your team is with Scrum, you may also want to look into training sessions. Certified Scrum Coaches and Trainers and Scrum Alliance Registered Education Providers can help your team learn and embrace Scrum.

Smartsheet's newest view, Card View, gives Agile teams a more highly-visual way to work, communicate, and collaborate in Smartsheet. Card View enables you to focus attention with rich cards, give perspective with flexible views, and prioritize and adjust work more visually. Display information on cards including custom fields, images, and color coding to better focus your team's attention. Categorize cards into lanes to organize your work more visually.

Use Smartsheet Card View during your next Scrum meeting.

Waterfall Methodology

What Is Waterfall?

Waterfall methodology follows a sequential, linear process and is the most popular version of the systems development life cycle (SDLC) for software engineering and IT projects. It is sometimes planned using a Gantt chart, a type of bar chart that shows the start and end dates for each task. Once one of the eight stages are complete, the development team moves onto the next step. The team can't go back to a previous stage without starting the whole process from the beginning. And, before the team can move to the next stage, requirements may need to be reviewed and approved by the customer.

The Waterfall model originated in the manufacturing and construction industries, both highly structured environment where changes can be too expensive or sometimes impossible. The first formal description of Waterfall is attributed to Winston W. Royce in a 1970 article where he described a flawed software model.

Advantages of Waterfall

Waterfall is best used for simple, unchanging projects. Its linear, rigid nature makes it easy to use and allows for in-depth documentation.

The advantages of Waterfall include:

Easy to use and manage: Because the Waterfall model follows the same sequential pattern for each project, it is easy to use and understand. The team doesn't need any prior knowledge or training before working on a Waterfall project. Waterfall is also a rigid model; each phase has specific deliverables and review, so it's easy to manage and control.

Discipline is enforced: Every phase in Waterfall has a start and end point, and it's easy to share progress with stakeholders and customers. By focusing on requirements and design before writing code, the team can reduce the risk of a missed deadline.

Requires a well documented approach: Waterfall requires documentation for every phase, resulting in better understanding of the logic behind the code and tests. It also leaves a paper trail for any future projects or if stakeholders need to see more detail about a certain phase.

Disadvantages of Waterfall

The biggest drawback of Waterfall is how it handles change. Because Waterfall is a linear, sequential model, you can't bounce between phases, even if unexpected changes occur. Once you're done with a phase, that's it.

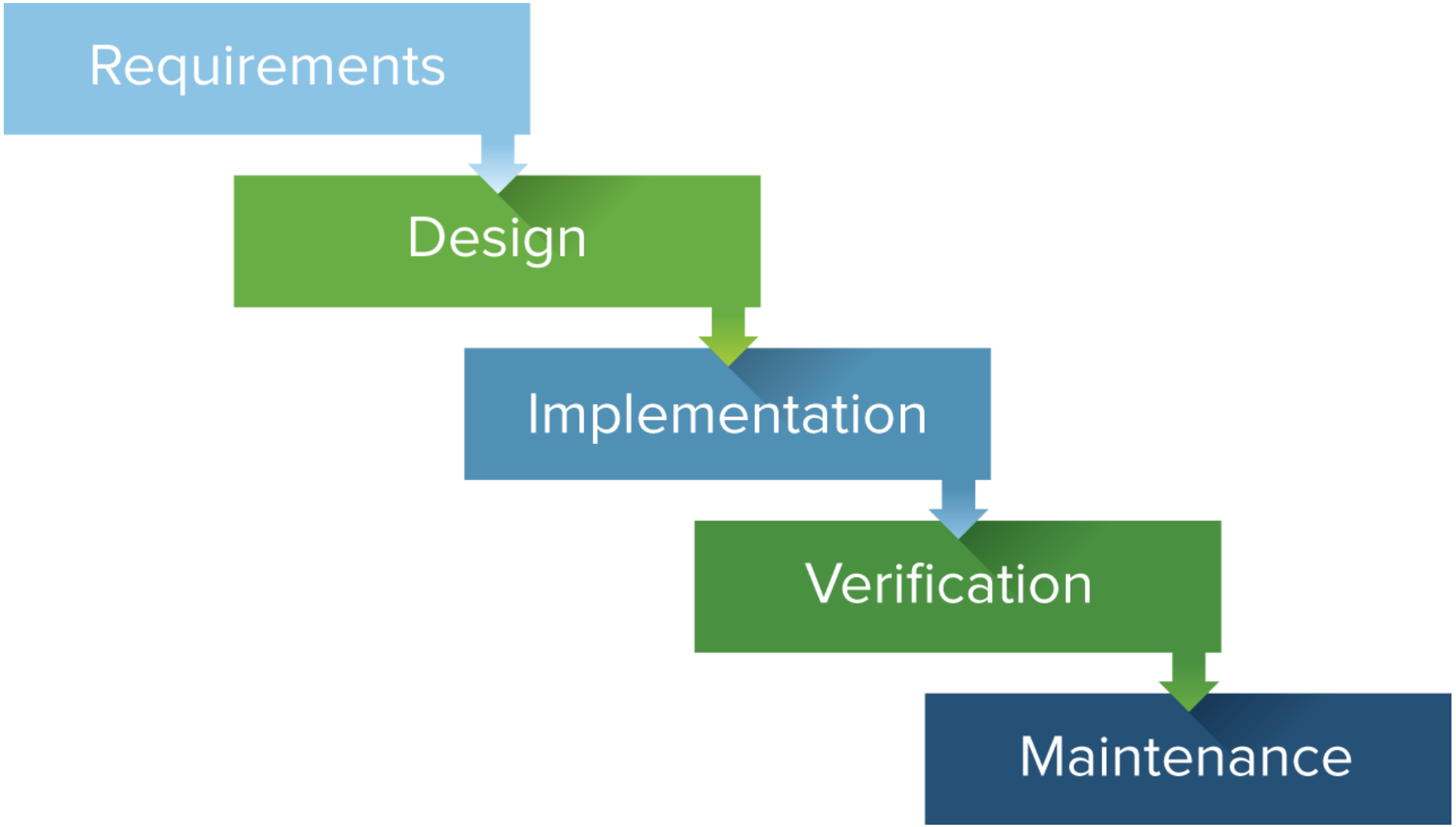
Here's more information on the disadvantages of Waterfall:

Changes can't be easily accommodated: Once the team completes a phase, they can't go back. If they reach the testing phase and realize that a feature was missing from the requirements phase, it is very difficult and expensive to go back and fix it.

Software isn't delivered until late: The project has to complete two to four phases before the coding actually begins. As a result, stakeholders won't see working software until late in the life cycle.

Gathering accurate requirements can be challenging: One of the first phases in a Waterfall project is to talk to customers and stakeholders and identify their requirements. However, it can be difficult to pinpoint exactly what they want this early in the project. Often times, customers don't know what they want early on and instead, learn and identify requirements as the project progresses.

Stages of Waterfall



There are eight stages in Waterfall and they must all happen in sequential order. For example, the development team can't go back to the analysis phase if they are in the testing phase.

Conception: This phase starts with an idea. The concept phase involves a rough assessment of the project, why it's beneficial, and looks at any initial cost estimates.
Initiation: Once the idea is formed, you need to hire the project team, and define objectives, scope, purpose, and deliverables.
Requirement Gathering and Analysis: Requirements are gathered and analyzed to see if the project is actually feasible. All this information is documented in a requirement specification document.
Design: The design specifications created in this phase are used in the coding phase to actually write the code. The requirements are studied and evaluated, and the design of the system is prepared. The team's goal is to understand what actions need to be taken and what they should look like.

Implementation/Coding: The actual coding of the software begins. Any flowcharts or algorithms created in the design phase are translated into a programming language.
Testing: Once the code is complete, the software needs to be tested for any errors. When the testing is finished, the software is delivered to the customer. Some teams may choose to include user acceptance testing (UAT), where users test the software before it is deployed to the general public.
Maintenance: Once customers have been using the software in the real world, they may find additional problems. The development team will need to resolve, change, or modify the software to continue to be effective.

Iterative Waterfall Development

In the traditional Waterfall model, the team goes through each phase for the entire project. For example, they do the analysis for the entire project, then they do the design for the entire project, etc.

In an iterative Waterfall model, there is still a lot of upfront planning required. Once the plan is in place, the team follows the same pattern as traditional Waterfall but does it for each story. They do the analysis for one story, then all the design for one story, then all the coding and testing for one story. Then they repeat the process for another story. The work is broken up into chunks that benefit the development team.

How Waterfall Deals with Software Requirements

Waterfall projects define all software requirements upfront. The project cannot proceed unless these requirements have been identified and documented.

Some Waterfall projects may have a dedicated team to capture, collect, and gather these requirements. They may use questionnaires, face-to-face or phone interviews, white boards, and modeling tools to capture stakeholder and customer requirements.

Once the initial requirements are defined, the team should produce a requirements specification document (sometimes they may create more than one). This document defines what needs to be delivered so everyone understands the scope of the project.

Kanban

What Is Kanban?

Kanban is Japanese for “visual sign” or “card.” It is a visual framework used to implement Agile that shows what to produce, when to produce it, and how much to produce. It encourages small, incremental changes to your current system and does not require a certain set up or procedure (meaning, you could overlay Kanban on top of other existing workflows).

Kanban was inspired by the Toyota Production System and Lean Manufacturing. In the 1940s, Toyota improved its engineering process by modeling it after how supermarkets stock shelves. Engineer Taiichi Ohno noticed that supermarkets stock just enough product to meet demand, optimizing the flow between the supermarket and customer. Inventory would only be restocked when there was empty space on the shelf (a visual cue). And because inventory matched consumption, the supermarket improved efficiency in inventory management.

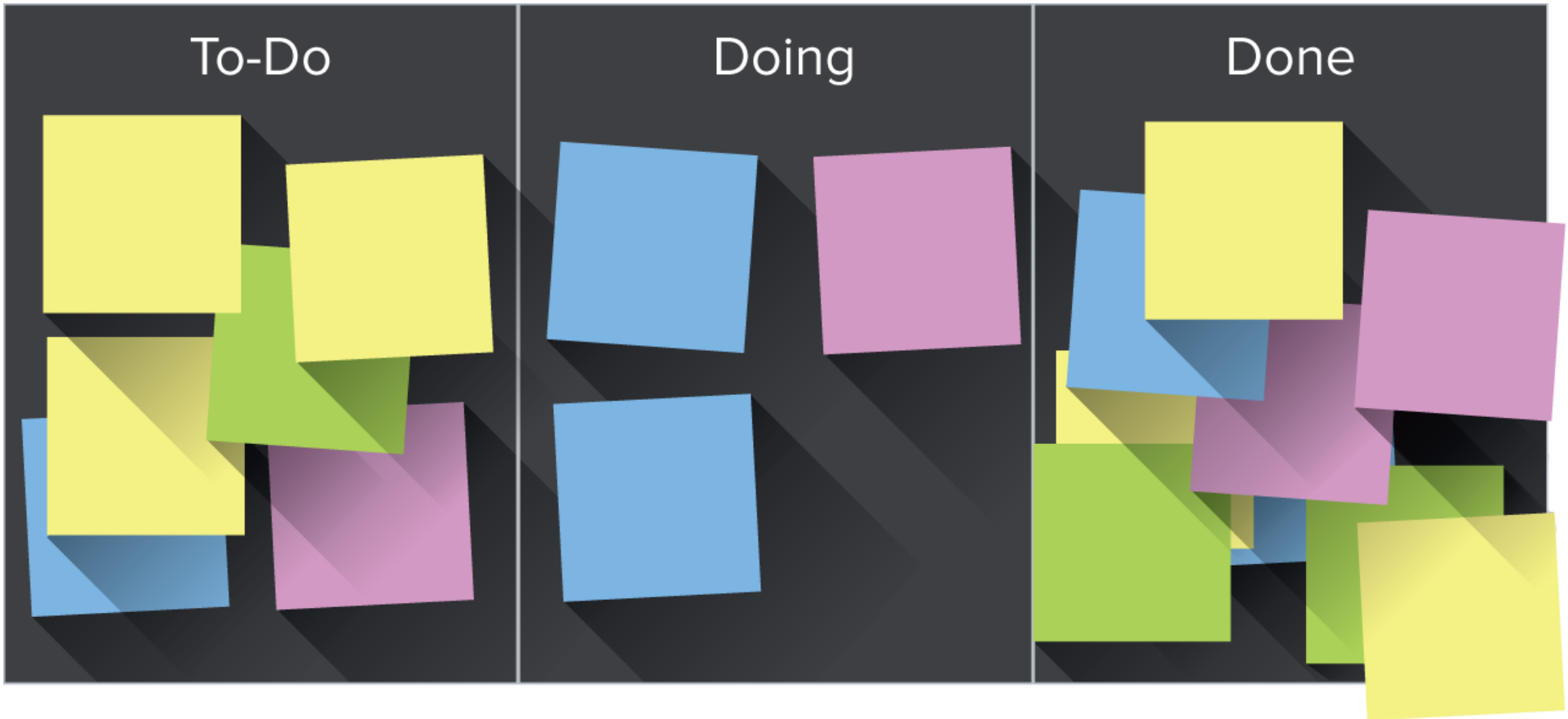
Toyota brought these same principles to its factory floors. Different teams would create a card (or Kanban) to communicate that they had extra capacity and were ready to pull more materials. Because all requests for parts were pulled from the order, Kanban is sometimes called the “pull system.”

These same ideas apply to software teams and IT projects today. In this context, development work-in-progress (WIP) takes the place of inventory, and new work can only be added when there is an “empty space” on the team’s visual Kanban board. Kanban matches the amount of WIP to the team’s capacity, improving flexibility, transparency, and output.

According to the Kanban blog (http://kanbanblog.com/), “Kanban is a technique for managing a software development process in a highly efficient way. Kanban underpins Toyota's ‘just-in-time’ (JIT) product system. Although producing software is a creative activity and therefore different to mass-producing cars, the underlying mechanism for managing the production line can still be applied.”

When looking at Kanban vs Agile, it’s important to remember that Kanban is one flavor of Agile. It’s one of many frameworks used to implement Agile software development.

About the Kanban Board



A Kanban board is a tool to implement the Kanban method for projects. Traditionally, this tool has been a physical board, with magnets, plastic chips, or sticky notes on a whiteboard to represent work items. However, in recent years, more and more project management software tools have created online Kanban boards.

A Kanban board, whether it is physical or online, is made up of different swim lanes or columns. The simplest boards have three columns: to do, in progress, and done. The columns for a software development project may consist of backlog, ready, coding, testing, approval, and done columns.

Kanban cards (like sticky notes) represent the work and each card is placed on the board in the lane that represents the status of that work. These cards communicate status at a glance. You could also use different color cards to represent different details. For example, green cards could represent a feature and orange cards could represent a task.

Advantages of Kanban

Kanban's visual nature offers a unique advantage when implementing Agile. The Kanban board is easy to learn and understand, it improves flow of work, and minimizes cycle time.

The advantages of Kanban include:

Increases flexibility: Kanban is an evolving, fluid model. There are no set phase durations and priorities are reevaluated as new information comes in.

Reduces waste: Kanban revolves around reducing waste, ensuring that teams don't spend time doing work that isn't needed or doing the wrong kind of work.

Easy to understand: The visual nature of Kanban helps to make it incredibly intuitive and easy to learn. The team doesn't need to learn a completely new methodology, and Kanban can be easily implemented on top of other systems in place.

Improves delivery flow: Kanban teams optimize the flow of work out to customers. Like continuous delivery (CD), Kanban focuses on the just-in-time delivery of value and delivering work to customers on a regular cadence.

Minimizes cycle time: Cycle time is the amount of time it takes for work to move through the team's workflow. In Kanban projects, the entire team helps to ensure the work is moving quickly and successfully through the process.

Disadvantages of Kanban

Many of the disadvantages associated with Kanban come with misuse or mishandling of the Kanban board. An outdated or overcomplicated board can lead to confusion, inaccuracies, or miscommunication.

Here's more on the disadvantages of Kanban:

Outdated board can lead to issues: The team must be committed to keeping the Kanban board up to date, otherwise they'll be working off inaccurate information. And once work is completed based off an out-of-date board, it can be hard to get things back on track.

Teams can overcomplicate the board: The Kanban board should remain clear and easy to read, however some team members may learn "new tricks" they can apply to their board. Adding these kinds of bells and whistles to the Kanban board just buries the important information.

Lack of timing: A frequent complaint about Kanban is that you don't know when things will be done. The columns on the Kanban board are only marked by phase (to do, in progress, complete), there are no timeframes associated with each phase, so you really don't know how long the to do phase could last.

Core Practices and Principles of Kanban

Every Kanban project should follow these core principles:

Visualize the workflow: A visual representation of your work allows you to understand the big picture and see how the flow of work progresses. By making all the work visible, including blockers and queues, you can identify issues early on and improve collaboration.

Limit work in progress (WIP): Work in progress limits (WIP limits) determine the minimum and maximum amount of work for each column on the board or for each workflow. By putting a limit on WIP, you can increase speed and flexibility, and reduce the need for prioritizing tasks.

Manage and enhance the flow: The flow of work (the movement of work) throughout the Kanban board should be monitored and improved upon. Ideally, you want a fast, smooth flow, which shows that the team is creating value quickly. The team should analyze problems in the flow then implement changes.

Make process policies explicit: In order for collaborative change to occur in the Kanban system, the processes need to be explicit. Everyone needs to understand how things work or what “done” really means. You can modify the board to make these processes more clear; for example, you could redesign it to specify how the work should flow.

Continuously improve: The Kanban method encourages small, continuous changes that stick. Once the Kanban system is in place, the team will be able to identify and understand issues and suggest improvements. Teams measure their effectiveness by tracking flow, measuring cycle time, and increasing quality of work.

Common Questions About Kanban

Q: How do you organize meetings and maintain focus without a Scrum Master?

Someone on the team needs to take initiative to put the meeting on the calendar and ensure the conversation stays on track. Even without a Scrum Master, it normally isn’t too big of an issue.

The Kanban board helps maintain focus during the meeting. During the meeting, you can go through the board from left to right and look for stories that have not moved since the last meeting. Instead of talking about accomplishments, you can just look at the cards on the board. The one question you do need to ask during a meeting is about the roadblockers or challenges to getting an item finished.

You could also try a kaizen meeting, where you only invite people who are involved in the task at hand. Each person discusses problems and challenges, and how his or her job could be done more efficiently. Then, the whole group talks about solutions to those issues.

Kaizen also can include a kaizen facilitator, who encourages the team to openly discuss critical issues.

Q: How can Kanban satisfy management’s desire for predictable delivery?

To some extent, Kanban trades predictability for efficiency. There are no timebox constraints or planning, however once a team has optimized the flow of work and can get a sense of how long certain tasks take, there will be some level of predictability.

If management still needs more defined predictability (which is not the Kanban approach), you may need to try managing expectations. In a traditional model, you have a predictable date of delivery, but in reality, no one is going to deliver a product by that date if it’s not complete. Management is always going to wait for the product to be complete, regardless of the original date set. In the Kanban model, the expectations need to be adjusted to focus on delivering the product when it’s ready and complete.

Q: How do you use Kanban when you’re on a deadline?

There are a couple different ways you can handle deadlines in a Kanban model. You can simply write the deadlines on the Kanban cards, making sure these deadlines act more as guidelines rather than hard-and-fast due dates (in Kanban, you shouldn’t sacrifice quality for timing).

You could also change how you and your team approach deadlines. In Kanban’s truest form, there is no need for them. The Kanban system will make sure that all tasks are completed as soon as possible, so a deadline is no longer necessary.

Q: Can Kanban be used for other projects besides software development?

Yes, Kanban can improve process results, reduce production times, and help manage workflow in almost any industry. For example, in the game development industry, Kanban helps shorten the video process timeline and reduce waste. In real estate, it brings more efficiency by tracking contracts, prospects, and listings on various boards. And in finance, Kanban can quickly identify bottlenecks and increase speed-to-market.

Q: Is WIP driven by resource availability?

Yes. When setting WIP limits, you need to look at how many people you have on your team and how many tasks you want them to work on at the same time.

Q: How do you know if the WIP limit is correct?

There is no formula for setting the right WIP limits. It's very common for limits to be wrong in the beginning, but you just need to adjust them as the project progresses. A good place to start is 1.5 for available resource, but you should constantly be reevaluating this number and making changes as necessary.

Agile vs Scrum

Differences and Similarities Between Agile and Scrum

	Scrum	Agile
Philosophy		X
Methodology	X	
Adds process	X	
Transparency	X	X
Deliver software early and often	X	X
Iterative	X	X
Accommodates change	X	X
Continuous improvement	X	X

While Agile and Scrum follow the same system, there are some differences when comparing Scrum vs Agile. Agile describes a set of principles in the Agile Manifesto for building software through iterative development. On the other hand, Scrum is a specific set of rules to follow when practicing Agile software development. Agile is the philosophy and Scrum is the methodology to implement the Agile philosophy.

Because Scrum is one way to implement Agile, they both share many similarities. They both focus on delivering software early and often, are iterative processes, and accommodate change. They also encourage transparency and continuous improvement.

How Does Scrum Fit with Agile?

Scrum is one of many frameworks used to implement an Agile process. Agile is an umbrella term that includes other processes, like Extreme Programming, Kanban, Crystal, and Scrum. Scrum is Agile, but Agile isn't Scrum.

When to Use Scrum

We recommend using Scrum if:

The project requirements will change and evolve
Continuous feedback is required
You have to figure out how to do a large part of the work because you haven't done it before
You don't need to commit to a fixed release date
The project team wants autonomy
You need to deliver software on a regular basis

Scrum works well for projects that have a lot of unknowns or that evolve over time. Scrum deals with these changes very effectively, so you can easily accommodate new information or features throughout the process.

When to Use Agile

The line between when to use Agile versus when to use Scrum is blurry. Scrum is one framework in the Agile process, so they both have a lot in common. A good place to start is to first understand if you should use Agile in general. Then, if an Agile methodology seems like it would work for you, you could choose which framework of Agile to use (Scrum being one framework).

We recommend using Agile if:

The final product isn't clearly defined
The clients/stakeholders need to be able to change the scope
Changes need to be implemented during the entire process
The developers are adaptable and can think independently
You need to optimize for rapid deployment

Hybrid Approach

If a pure Scrum approach doesn't work for your project, you can also try a hybrid model. There are several methodologies that combine the principles of Agile or Scrum and adapt the framework to scale more effectively.

For example, Disciplined Agile Delivery (DAD) builds on the practices of Agile, Scrum, and Lean to provide a solid foundation from which to scale. DAD was developed to provide a more cohesive approach to Agile, taking strategies from Scrum, Kanban, Extreme Programming, and others. Rather than taking the time to learn one of these existing frameworks and cobble them together as needed, DAD already combines all relevant techniques.

Other hybrid methods include Large-Scale Scrum (LeSS), which extends Scrum with scaling rules and guidelines, and Scaled Agile Framework (SaFE), based on underlying Lean and Agile principles.

Kanban vs Scrum

Differences and Similarities: Scrum vs Kanban

	Scrum	Kanban
Specific roles	X	
Timeboxed iterations	X	
Accomodates change		X
Estimation	X	
Empirical	X	X
Lean and agile	X	X
Limits WIP	X	X
Work can be done simulatenously	X	X
Board is continuously used		X
Teams must be cross functional	X	
Pull scheduling	X	X
Transparency	X	X
Deliver software early and often	X	X

Scrum and Kanban are both flavors of Agile, but they have some distinct differences.

Scrum requires **specific roles** whereas Kanban has no required roles.

Scrum is based on **timeboxed iterations**, combining planning, process improvement, and release. In Kanban, you can choose to do these activities on a regular cadence or whenever you need.
Scrum limits **work in progress (WIP)** in each iteration, whereas Kanban limits WIP in each workflow.
Scrum **resists change**, whereas Kanban easily accommodates and embraces change. In Scrum, once the team has committed stories to a sprint, you can't add additional stories later on. In Kanban, you can add or change stories as you please, assuming that it's within WIP limits.
A **Scrum board is reset** after each sprint. A Kanban board is continuously used.
A Scrum team is **cross-functional** and one team owns the Scrum board. In Kanban, teams don't need to be cross-functional and anyone can own the Kanban board.
Scrum teams require **estimation**, whereas Kanban doesn't.

And Scrum and Kanban also have some similarities:

They are empirical. You have to experiment with the process to see what works for you.
Both allow team members to work on multiple products at once.
They are Lean and Agile.
They both limit WIP (although the way they each limit WIP is different)
They use pull scheduling
They focus on delivering software early and often
Both use transparency to improve process

How Do Kanban and Scrum Relate to Each Other?

Kanban and Scrum are both frameworks for Agile software development. They both take large, complex tasks and break them down into smaller chunks. Kanban and Scrum also work toward continual improvement and optimization of the process, and want to keep work highly visible.

While both Kanban and Scrum are very adaptive, Scrum is more rigid than Kanban. Scrum has more constraints, whereas Kanban is more flexible.

Scrum Board vs Kanban Board

While a Scrum board and Kanban board can look similar visually, they are based on very different principles.

To create a Scrum board, the Scrum team must first create sprints, assign points to user stories, and plan which stories go into which sprint. Then, the Scrum board visualizes the sprint, showing which stories are in plan mode or work mode. The Scrum board is reset between each sprint and is owned by one specific team.

A Kanban board has the same column-based layout as a Scrum board, but it requires no upfront planning. You can start working and moving through the flow of the Kanban board without having a structured plan. The Kanban board can be shared by multiple people and is persistent; you don't need to reset the board. And, unlike the Scrum board, the Kanban board has a maximum number of stories allowed in each column at one time. This will continue to flow as long as the project continues, with new stories added and completed stories being reevaluated if needed.

When to Use Kanban

We recommend using Kanban if:

You need to add stories or change sprints on the fly
You don't need iterations
Estimation isn't necessary

You want the ability to release at any time
Continuous improvement is already emphasized
Your team doesn’t respond well to big changes
You want to improve delivery flow
The system needs to be easy to understand

Scrum can be less flexible than Kanban. The timing revolves around sprints, with each sprint lasting two to four weeks. In each sprint, the team has specific roles and follows specific ceremonies.

What Is Scrumban?

Scrumban combines the principles of Scrum and Kanban into a pull-based system. The team plans out the work that was established during initiation and continually grooms the backlog. The same Scrum meetings should take place, but the frequency can change depending on context and need. The most important part of Scrumban is making sure that work in progress limits (WIP limits) are followed.

Scrumban takes bits and pieces from both Scrum and Kanban. For example, it includes the defined roles, daily Scrum, and other meetings from Scrum. And from Kanban, it takes the Kanban board, continuous flow, and ability to add changes as needed to the board.

Scrumban can look more like Scrum on the technical level, but at the cultural level, it will more closely resemble Kanban. Instead of big changes all at once, Scrumban encourages incremental changes. If your team is looking to migrate from Scrum to Kanban, Scrumban can provide a gentle transition.

Which One Is Best? Kanban vs Scrum

When comparing Kanban versus Scrum, there is no definitive winner. The best framework depends on your project, team, and your goals. Because both Kanban and Scrum are flexible Agile methodologies, you could easily take principles from each and apply them as you see necessary.

It’s important to remember that true Scrum is a much bigger shift than Kanban. The team will have to learn about the ceremonies, the specific roles, and iterations. On the other hand, Kanban encourages incremental improvements. You can apply Kanban principles to any process you already have in place, even Scrum. Nothing needs to change significantly to get started with Kanban.

As a general rule of thumb, if your team or organization is really stuck and needs a big change, Scrum may be more appropriate. If you already have a process in place that you’re happy with, but want to implement some small changes, Kanban might be a better choice.

Agile vs Waterfall

Differences and Similarities: Waterfall vs Agile

	Waterfall	Agile
Sequential	X	
Flexible		X
Accommodates change		X
Defined requirements	X	
Deliver quality products	X	X
Continually evolving		X
Rigid process	X	

The differences between Waterfall methodology versus Agile can be summed up in two words: rigid vs flexible. Waterfall is a much stricter, rigid process whereas Agile is flexible and continuously evolving. Here's more on their differences:

Waterfall is a **structured** process, where you can't start on a new phase until the previous one has been completed. On the other hand, Agile is a flexible process, allowing you to move through the project as you like.
Waterfall is **sequential** and Agile does not enforce a linear process.
Waterfall projects usually include **defined requirements** in advance, whereas requirements are expected to change and evolve in Agile projects.
In Waterfall projects, you can't **change** things that were done in previous stages, whereas Agile is very accommodating to changes.

There are not many similarities between Agile and Waterfall; Agile was specifically created to be the opposite of Waterfall. However, you can say that both Agile and Waterfall have the same goal. They both want to deliver quality products in an efficient way. If you have any other similarities between Agile and Waterfall to share, please leave us a comment!

When You Should Use Waterfall and When to Use Agile

We recommend using Waterfall if:

You don't expect changes in scope and you're working with fixed-price contracts
The project is very simple or you've done it many times before
Requirements are very well known and fixed
Customers know exactly what they want in advance
You're working with orderly and predictable projects

And you should use Agile if:

The final product isn’t clearly defined
The clients/stakeholders need the ability to modify the scope
You anticipate any kind of changes during the project
Rapid deployment is the goal

When deciding between Agile versus Waterfall, it can all boil down to this: if you anticipate or expect any changes throughout the project, go with Agile. If you know the project is fixed, unchanging, and predictable, Waterfall may be a better choice.

Which One Is Better? Agile vs Waterfall

Agile and Waterfall are such opposites that it’s hard to say which one is better. It really depends on the project, the level of clarity around requirements, and how flexible you can be.

If you have a clear picture of what the final product should be, you have fixed requirements that won’t change, and you’re working on a relatively simple project, some argue that Waterfall is a better choice than Agile. If you don’t expect to deal with change, Waterfall is a straightforward, efficient process. The issues with Waterfall come when you have to accommodate changes.

If you don’t have a clear picture of the final product, you anticipate changes, and you’re working on a complex project, Agile is superior. Agile is designed to accommodate new, evolving requirements any time during the project, whereas Waterfall does not allow you to go back to a completed phase and make changes.

Hybrid: Agifall or WAgile

If you’re still wondering about Waterfall versus Agile, you could always combine principles of both and use a hybrid model.

Agifall, for example, increases speed and quality by adding Agile methodologies to the Waterfall process. In an Agifall project, you would break out the research, strategy, and planning phases into tasks and proceed with sprints to complete them. The development phase would be just like any other Agile project, with more information up front. You also don’t need to wait for one phase to end to start the following phase, which is traditional in pure Waterfall. With Agifall, when the project can begin, it should begin.

Wagile has a more negative connotation than Agifall. The definition of Wagile on Wikipedia is, “a group of software development methodologies that result from slipping from Agile back into Waterfall, doing a lot of short Waterfalls and thinking it is Agile, Waterfall model masquerading as Agile software development.”

Wagile adopts Agile practices like short iterations, daily stand-ups, or continuous integration on top of the Waterfall model, without really changing the traditional Waterfall model.

Kanban vs Agile

Differences and Similarities: Agile vs Kanban

	Kanban	Agile
Continuous flow	X	
Iterations		X
Visualization	X	
Cross-functional teams		X
Equally beneficial for all industries	X	
Philosophy		X
Continuous improvement	X	X
Transparency	X	X
Don't require upfront planning	X	X
Faster delivery	X	X
Breaks projects into smaller chunks	X	X

While Kanban is a visual way to implement Agile, they have many differences:

Kanban advocates for **continuous flow**, while Agile works in iterations.
Kanban can work equally well for **any type of work**, whereas Agile may be better suited for some projects rather than others.
Anyone can pick up Kanban, but some Agile methodologies require **knowledge or training**.
Kanban requires a **visual representation** of workflow, while Agile does not.
Some Agile projects require **cross-functional teams**, whereas Kanban does not.
Agile is a philosophy whereas Kanban is a method.

And Agile and Kanban also have similarities:

They both break down projects into smaller chunks.
They emphasize continuous improvement.
They place high value on transparency.
Neither of them require a lot of upfront planning.
They work toward faster delivery.

Kanban vs Agile

When You Should Use Kanban and When to Use Agile

We recommend using Kanban if:

Your project doesn’t require iterations
You want the ability to release at any time
Your team prefers incremental change
Your team works well with visuals
You want to improve delivery flow
You’re looking for an easy-to-understand system

And we recommend using Agile if:

The final product isn’t clearly defined
Changes need to be implemented during the entire process
The developers are adaptable and can think independently
You’re looking to make a substantial change

Kanban vs Agile

Which One Is Better? Agile vs Kanban

Like with any project management methodology, there isn’t one framework that is better 100% of the time. You may choose Kanban for some projects, but want to implement Agile for others.

Consider what level of change you want to introduce to your team. If you want to add something on top of an existing framework with small, incremental changes, Kanban is a better choice. If you’re looking to make a bigger process change, implementing Agile (like Scrum) would be better.

And, if you want your project team to get started right away with a new method, Kanban is easier to understand. There is no training required and it can be used on top of any existing process. On the other hand, some Agile methods require more knowledge from the team. For example, they may need to learn specific roles, ceremonies, and terminology.

More Resources

Resources and Related Posts

How to Create a Waterfall Chart in Excel (https://www.smartsheet.com/how-create-Waterfall-chart-excel)

Download a free Excel waterfall chart template or learn how to create a waterfall chart from scratch. We'll also share when to use a waterfall chart and the features of a waterfall chart in Excel.

Best Agile Project Management Excel Templates (https://www.smartsheet.com/Agile-project-management-excel-templates)

Find eight Agile project management templates in Excel, ranging from Agile product backlog template to Agile project charter template. You'll also learn how to use Agile templates in Smartsheet

Agile Planning: Best Practices for Project Managers (https://www.smartsheet.com/best-practices-guide-agile-planning-project-managers)

Agile One-Stop Project Management Resource (https://www.smartsheet.com/everything-you-need-to-know-about-agile-project-management)

Courses:

PMI Agile Certified Practitioner (PMI ACP) (https://www.pmi.org/certification/agile-management-acp.aspx): Offered by the Project Management Institute (PMI), this certification covers the many different approaches to Agile, like Scrum, Kanban, Lean, Extreme Programming (XP), and Test-Driven Development (TDD). Prerequisites include 2,000 hours of general project experience working on a team, 1,500 hours working on Agile project teams, and 21 contact hours of training in Agile practices.
Certified ScrumMaster (CSM) (https://www.scrumalliance.org/certifications/practitioners/certified-scrummaster-csm): This certification from Scrum Alliance helps teams properly use Scrum, understand the values, and protect the team from distractions. As a CSM, you will be able to fill the role of Scrum Master or Scrum team member. To earn your CSM certificate, you must take a CSM course from a Scrum Alliance Authorized Trainer and demonstrate progress with an online test.
Certified Scrum Product Owner (CSPO) (https://www.scrumalliance.org/certifications/practitioners/cspo-certification): A Certified Scrum Product Owner learns Scrum terminology, practices, and principles to fulfill the role of Product Owner on a Scrum team. He or she is closest to the business side of the project, maintains the product backlog, and ensures everyone knows the priorities. To earn this certification from the Scrum Alliance, you must attend an in-person, two-day CSPO course taught by a Certified Scrum Trainer.
Certified Scrum Developer (CSD): (https://www.scrumalliance.org/certifications/practitioners/csd-certification) Certified Scrum Developers learn specialized Agile engineering skills and demonstrate their knowledge through formal training and a technical skills assessment. The CSD course is geared toward software developers who are working in a Scrum environment. To earn a CSD from the Scrum Alliance, you must have five days of formal training taught by a Scrum Alliance Registered Education Provider and a Scrum Alliance Authorized Instructor.
Certified Scrum Professional (CSP): (https://www.scrumalliance.org/certifications/practitioners/csp-certification) Certified Scrum Professionals challenge their Scrum teams to improve the way Scrum is implemented for every project. To apply for a CSP, you must currently hold a CSM, CSPO, or CSD credential, have a minimum of 36 months of Agile/Scrum experience, and gather and submit 70 Scrum Education Units from the past three years.
Accredited Kanban Practitioner (AKP) (http://www.agilecertifications.org/akp.php): Accredited Kanban Practitioners are professionals who have proven knowledge and expertise in Kanban implementation for software development. The certification is offered by the Agile Certification Institute, Inc. and requires that you have prior training in Agile practices and that you pass an AKP certification exam.

Agile Project Management with Smartsheet

Manage Any Project Your Way with Smartsheet

Smartsheet is a work management tool with powerful collaboration and communication features. By providing a broad range of smart views – Grid, Calendar, Gantt, Dashboards – Smartsheet helps you manage projects the way you want to.

Its pre-built project management templates make it even easier to track project requirements, store documents, create timelines, and organize key details. You can make real-time updates and alert your team about the new changes, and share your plan with internal and external stakeholders to increase transparency and keep everyone on the same page.

Plus, with our newest view, Card View, teams have a more visual way to work, communicate, and collaborate in Smartsheet. Card View enables you to focus attention with rich cards, give perspective with flexible views, and prioritize and adjust work more visually.

Display information on cards including custom fields, images, and color coding to better focus your team's attention. Categorize cards into lanes to organize your work more visually. Intuitively change lanes and filter cards to see the flow of work from multiple perspectives. Act on tasks and change status of work by dragging and dropping cards through lanes to immediately share decisions with the entire team. Start with a pre-built Card View template or import existing projects directly from Trello.

Manage Your Agile Project in Smartsheet (https://www.smartsheet.com/s/software-development-solution-signup?lpv=butbot&trp=8586&lx=Ta7o6Ol-r8vCMohDULSCsg)

Want more project management tips and best practices? Don't miss our Project Management Resource Hub (https://www.smartsheet.com/resources/project-management) for the latest articles, templates, videos, and more.

*The PMBOK Guide is a registered mark of the Project Management Institute, Inc.

Comments

Using Smartsheet with Kanban

Submitted by Peter on Fri, 05/20/2016 - 09:04

Great post! Are there any templates available in Smartsheet to support Kansan / Agile? Will you release the CardView in Smarsheet in the next release?
reply (https://www.smartsheet.com/comment/reply/5180/43211)

RE: Using Smartsheet with Kanban

Submitted by Smartsheet Comm... on Tue, 05/24/2016 - 15:06

Hello Peter, CardView is in development and coming soon. Stay tuned for news on the release! https://www.smartsheet.com/product-roadmap
reply (https://www.smartsheet.com/comment/reply/5180/43329)

Apples to Oranges to Cherries

Submitted by Concerned Citizen on Tue, 08/09/2016 - 06:43

Agile is a philosophy, Scrum is a framework, Waterfall is a method, Kanban is a control system. While there are some links between them in history and intent, they should not be directly compared. Some information is inaccurate. It is good that the Manifesto is linked. The Scrum Guide (scrumguides.org) is the authoritative document and should be linked. The intent is appreciated, but the result is lacking.

reply (https://www.smartsheet.com/comment/reply/5180/45976)
Thanks for such a

Submitted by Sarabeth Marcello on Tue, 09/20/2016 - 09:57

Thanks for such a comprehensive post--you've answered pretty much all my questions about project management methodologies. I'll definitely be referring back to this post in the future. Do you have any plans to create a shorter guide with the most important ideas for quick reference?
reply (https://www.smartsheet.com/comment/reply/5180/47173)
Hi, thank you for this post I

Submitted by tushar on Mon, 02/27/2017 - 20:55

Hi, thank you for this post I agree with you that If you have a clear picture of what the final product should be, you have fixed requirements that won't change, and you're working on a relatively simple project, some argue that Waterfall is a better choice than Agile.very useful information
reply (https://www.smartsheet.com/comment/reply/5180/66361)
Add new comment

Your name

Subject

Comment *

Our Privacy Policy (legal/privacy) describes how we process your personal data.

By submitting this form, you accept the Akismet privacy policy (http://automattic.com/privacy/).

(http://www.facebook.com/smartsheet) (https://www.linkedin.com/company/smartsheet-com) (https://www.youtube.com/user/SmartSheet)

Contact Us (https://www.smartsheet.com/contact?fts=contact-footer)

844.324.2360 (tel:18443242360)

Smartsheet (https://www.smartsheet.com/why-smartsheet)

About Us (https://www.smartsheet.com/about)

Management (https://www.smartsheet.com/about/management)

Board of Directors (https://www.smartsheet.com/about/board-of-directors)

Blog (https://www.smartsheet.com/blog)

Customers (https://www.smartsheet.com/customers)

Newsroom (https://www.smartsheet.com/news)

Trust Center (https://www.smartsheet.com/trust)

Channel Partners (https://www.smartsheet.com/channel-partners)

Careers (https://www.smartsheet.com/careers)

ENGAGE '18 (https://www.smartsheet.com/engage)

Investors (https://investors.smartsheet.com)

Product (https://www.smartsheet.com/product)

Smartsheet Overview (https://www.smartsheet.com/product)

Make Collaboration Work (https://www.smartsheet.com/product/collaborate)

See More, Manage More (https://www.smartsheet.com/product/manage)

Automate Work Processes (https://www.smartsheet.com/product/automate)

Deploy with Confidence (https://www.smartsheet.com/product/enterprise-deployment)

Work at Scale with Control Center (https://www.smartsheet.com/product/control-center)

Apps and Integrations (https://www.smartsheet.com/partners)

Developers & API (http://developers.smartsheet.com/)

Feature Overview (https://www.smartsheet.com/feature-overview)

Converse.AI (http://www.converse.ai/)

Solutions (https://www.smartsheet.com/solutions)

Business Solutions (https://www.smartsheet.com/solutions#Business-Solutions)

By Industry (https://www.smartsheet.com/solutions#Solutions-By-Industry)

By Role (https://www.smartsheet.com/solutions#Solutions-By-Function)

Support (https://www.smartsheet.com/help-center)

Help and Learning Center (https://www.smartsheet.com/help-center)

Community (https://community.smartsheet.com)

User Groups (https://www.smartsheet.com/user-groups)

Contact Support (https://help.smartsheet.com/contact)

Status (https://status.smartsheet.com/)

Legal (https://www.smartsheet.com/legal)

Terms of Use (https://www.smartsheet.com/legal)

Privacy (https://www.smartsheet.com/legal/privacy)