The Project Management Blueprint Part 1: A Comprehensive Comparison of Agile, Scrum, Kanban, and Lean

DARREN HAGMAN
Darren is a veteran Scrum master with experience in Waterfall and Agile across a number of industries.

Overview

Many methodologies are used in software development today. You may have heard buzzwords such as Waterfall, Agile, Scrum, Kanban, Lean, Extreme Programming, etc.

In this article, I will define these terms, discuss how they are related to each other and how they differ from each other. Many of the aforementioned buzzwords are based on concepts from Lean Manufacturing which was originally based on the Toyota Manufacturing System (TPS) so we will talk about this first.

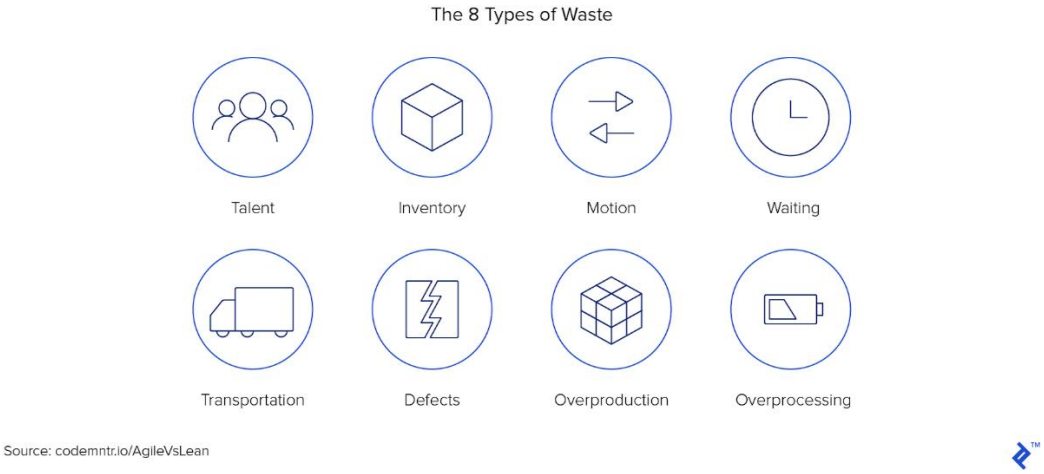
Lean Methodology

Origins of Lean and Lean Manufacturing

The term “Lean” has its origins in manufacturing where it was coined to describe a manufacturing model based on the Toyota Production System (TPS) originally developed by Sakichi Toyoda, Kiichiro Toyoda, and Taiichi Ohno who were originally inspired by Henry Ford. TPS was focused on the philosophy of “the complete elimination of all waste” and revolutionized manufacturing in the 1950s thru 1970s. TPS become known as “Lean Manufacturing” in 1990 when “The Machine That Changed the World” was published.

TPS identified three broad types of waste at Toyota:

- **Muda:** translated as “waste.” There were seven types of muda identified at Toyota and an eighth was later added. These are:
    - **Defects:** effort involved in finding and fixing defects
    - **Overproduction:** production ahead of demand
    - **Waiting:** waiting for the next production step, interruptions, etc.
    - **Non-used Talent:** underutilizing capabilities, inadequate training, etc
    - **Transport:** moving parts or products that aren’t required for processing
    - **Inventories:** finished inventory and work in progress
    - **Motion:** moving or walking more than what is needed for processing
    - **Excess Processing:** from poor tooling or product design



- **Muri:** translated as “overburden.” Muri usually results from mura but can result from muda. Muri manifests itself in break-downs, absenteeism, safety issues, etc.
- **Mura:** translated as “unevenness.” Mura can be found in fluctuation in customer demand, process times per product or variation of cycle times for different operators. When mura is not reduced, one increases the possibility for muri and, therefore, muda. Mura can be reduced by creating openness in the supply chain, changing product design, and creating standard work for all operators.

TPS worked to eliminate waste by applying these two core concepts:

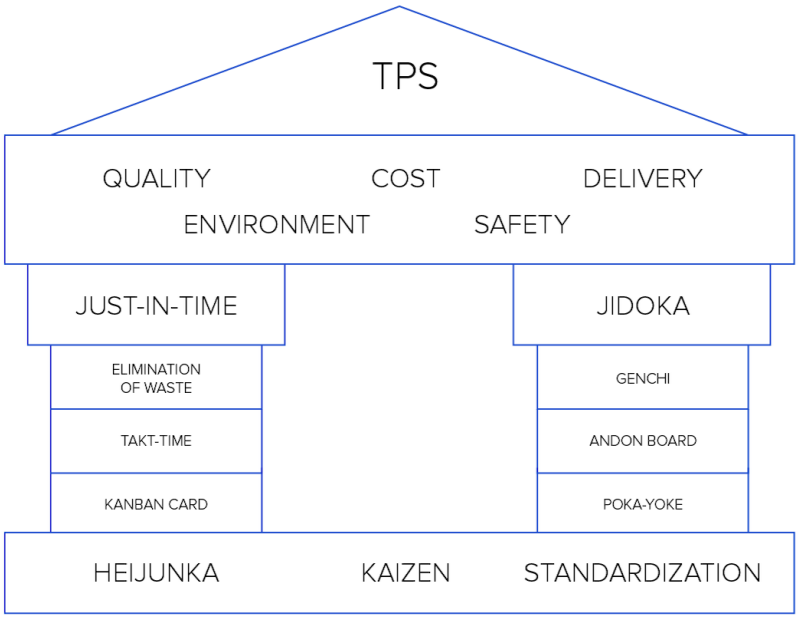
- **Jidoka:** loosely translated as “automation with a human touch” stipulates that “Quality must be built in during the manufacturing process!” which means that when a problem occurs, the equipment stops immediately, preventing defective products from being produced.
- **Just-in-Time:** Making only “what is needed, when it is needed, and in the amount needed.”

As TPS evolved, these core pillars and values built upon the concepts of Jidoka and JIT and became entrenched:

- Continuous Improvement:
    - **Challenge:** forming a long-term vision and meeting challenges with courage and creativity to realize dreams

    - **Kaizen:** improving business operations continuously, always driving for innovation and evolution, eliminating one muda at a time
    - **Genchi Genbutsu:** practicing genchi genbutsu, going to the source to find the facts to make correct decisions, build consensus, and achieve goals at our best speed
- Respect for People:
    - **Respect:** respecting others and making every effort to understand each other, taking responsibility and doing our best to build mutual trust
    - **Teamwork:** stimulating personal and professional growth, sharing opportunities for development, and maximizing individual and team performance
- **Andon:** a visual indicator of status or trouble
- **Heijunka:** means leveling or production leveling
- **Hansei:** means self-reflection. To improve efficiency, workers should challenge assumptions behind current processes to find better ones.
- **Kanban:** a signboard used as a visual tool to control production
- **Poka-yoke:** Also referred to as mistake proofing or error proofing
- **Pull System:** Material is pulled into a workstation just as it is needed
- **Seiri:** is the principle that mirrors waste. Seiri dictates that what is unnecessary should be removed. This encompasses all of the original seven wastes of TPS
- **Standardization:** organizes all jobs around human motion and creates an efficient production sequence without muda. This helps lead to quality, a constant pace, and allows for continuous improvement.

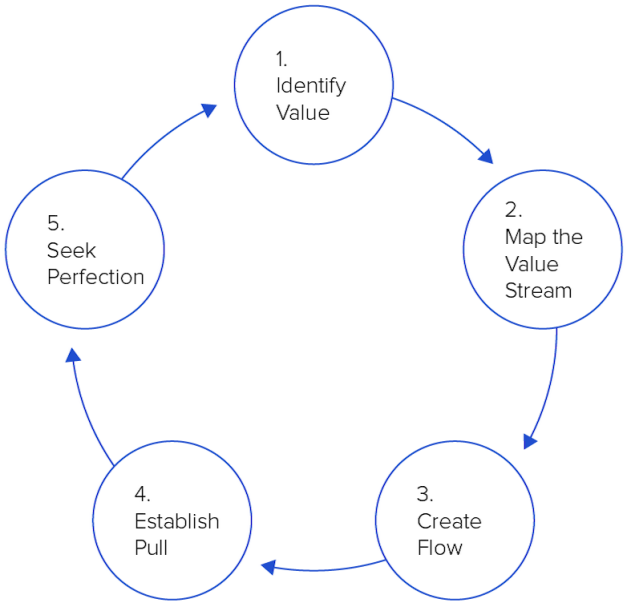
The diagram below shows how the core concepts and core values relate to each other.



Lean Management

The Toyota Product System and Lean Manufacturing evolved over time and were applied in a number of areas including management.

Lean Management applied the core values of continuous improvement and respect for people and distilled it into a set of five prescriptive Lean principles which would be repeated a number of times to continuously improve and eliminate waste:



1. **Identify Value:** Specify a value from the standpoint of the end customer by product family.
2. **Map Value Stream:** Identify all the steps in the value stream for each product family, eliminating whenever possible those steps that do not create value.
3. **Create Flow:** Make the value-creating steps occur in tight sequence so the product will flow smoothly toward the customer.
4. **Establish Pull:** As flow is introduced, let customers pull value from the next upstream activity.
5. **Seek Perfection:** As value is specified, value streams are identified, wasted steps are removed, and flow and pull are introduced, begin the process again and continue it until a state of perfection is reached in which perfect value is created with no waste.

These principles and other aspects of Lean management were formalized when Womack & Jones published “Lean Thinking” in 1996.

Lean Software Development

Lean has since been applied to management, software development, and other fields.

In the 1980s and 1990s, the software development industry was approaching a crisis as projects executed using traditional waterfall methodologies were taking longer and longer. This often resulted in a huge lag between a business need being identified and a software solution being delivered. Sometimes this lag was measured in multiple years, or even in decades in certain industries with specific requirements, such as the aerospace industry.

During these multiyear timeframes, requirements, systems, or even entire businesses changed. Often, projects would be canceled part way through or they would complete their scope, only to find out that what they delivered no longer meets the business needs as identified at the beginning of the project.

The Waterfall methodology rewarded stakeholders for thinking of everything as they were forced into writing a contract even though they probably weren't sure of what they needed. The Waterfall methodology forced decisions to be made during the requirements or design phase, and these decisions were very hard to change without changing the contract and adding costs to the project. A high percentage of software projects failed completely, or delivered late and over budget, or failed to deliver what was needed.

This general frustration led to various thought leaders trying to improve on Waterfall. Early examples include Rapid Application Development (RAD) which focused on reducing waste by shortcutting the requirements and design phases via rapidly developing a prototype and collaborating with business to further develop the requirement. There was also a move toward iterative development where a small project was completed and features were added in continual iterations. While these methodologies helped, they did not solve the core problems associated with Waterfall.

In the 1990s and early 2000s, several authors published books on applying Lean principles to software development. Dr. Robert Charette published "Lean Software Development" in 1993 and "12 Principles of Lean Software Development" in 2003. Tom and Mary Poppendieck published "Lean Software Development: An Agile Toolkit" in 2003. This book detailed seven principles of Lean Development, which correlates directly to the seven forms of waste in Lean Manufacturing. The similarities and differences between the two different Lean publications and Agile (discussed in the next section) are laid out in the diagram below.

DIFFERENCES BETWEEN LEAN AND AGILE

According to Dr. Charette, one of the primary differences between Lean and Agile is that Agile is bottom up, while Lean is top down.

Goals

Charette's Lean Software Development	The Agile Manifesto	Poppendieck's Lean
1. 1/3 Human effort 2. 1/3 Development hours 3. 1/3 Time 4. 1/3 Investment 5. 1/3 Effort to adapt	1. Individuals and interactions 2. Working software 3. Customer collaboration 4. Responding to change	

Principles

Charette's Lean Software Development	The Agile Manifesto	Poppendieck's Lean
1. Customer satisfaction 2. Value for money 3. Customer participation 4. Team effort 5. Everything is changeable 6. Domain, not point solutions 7. Complete, don't construct 8. 80% Solution today 9. Minimalism is essential 10. Needs determine tech	1. Customer satisfaction 2. Welcome changing requirements 3. Frequent cycles of delivery 4. Stakeholder collaboration 5. Culture of trust, support, and motivation 6. Face-to-face communication 7. Working software is the metric 8. Sustainable development 9. Technical excellence 10. Simplicity	1. Eliminate waste 2. Amplify learning 3. Deliver as fast as possible 4. Decide as late as possible 5. Empower the team 6. Build integrity in the product 7. See the whole process

Charette's Lean Software Development	The Agile Manifesto	Poppendieck's Lean
11. Product growth is feature growth 12. Mind the limits	11. Self-organizing teams 12. Team reflection	

Agile Framework

Origins of Agile and The Agile Manifesto

Around the same time that Charette and the Poppendiecks published their books, the Agile Framework was created to help solve the same problems. In February 2001, a group of Agile pioneers met at the infamous Snowbird meeting in Snowbird, Utah to try and come up with a solution.

The result was the Agile Manifesto which laid out a set of values and principles for a methodology that attempts to adapt to changing requirements and customer needs, cut waste, and deliver benefits faster using an incremental, iterative approach.
The Agile Manifesto reads as follows:

“We are uncovering better ways of developing software by doing it and helping others do it. 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

That is, while there is value in the items on the right, we value the items on the left more."

Aligned with the values in the manifesto are the 12 principles behind the Agile Manifesto:

"We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress. Agile processes promote sustainable development.

8. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

The above values and principles are applications of Lean principles such as Jidoka, JIT, Genchi Genbutsu, Kaizen, Hansei, Heijunka, and reducing waste.

Agile Principles Applied to Software Development

Agile is an umbrella framework that applies to any process that applies the Agile set of values and principles.

Some examples are:

- Extreme Programming
- Scrum
- Kanban

Scrum

Brief History of Scum

Scrum is a framework that applies Agile principles that was invented separately by multiple people, several of whom signed the Agile Manifesto:

- Hirotaka Takeuchi and Ikujiro Nonaka initially introduced the term “scrum” in a manufacturing context in their white paper “The New Product Development Game.” published in 1986 in the Harvard Business Review.
- Jeff Sutherland, John Scumniotales and Jeff McKenna implemented Scrum at the Easel Corporation in 1993.
- Ken Schwaber used what would become Scrum at his company, Advanced Development Methods in the 1990s .

Schwaber and Sutherland collaborated throughout the 1990s to develop and refine the framework in a software development context, speaking at various conferences. Schwaber worked with Mike Beedle to describe the method in the book, “Agile Software Development with Scrum” in 2001.

Schwaber went on to create both of the main Scrum certification authorities:

- Scrum Alliance: created in 2001. Set up the **Certified Scrum** accreditation series.
- Scrum.org: created in 2009 after Schwaber left the Scrum Alliance. Set up the parallel **Professional Scrum** accreditation series.

Over time, several frameworks/certification bodies were created to address scaling of the Scrum framework to larger teams and projects as Scrum was originally designed for small teams (7 plus or minus 2 members):

- SAFe: Scaled Agile Framework

- LeSS: Large Scale Scrum
- Scrum@scale: Scrum at Scale created by Jeff Sutherland

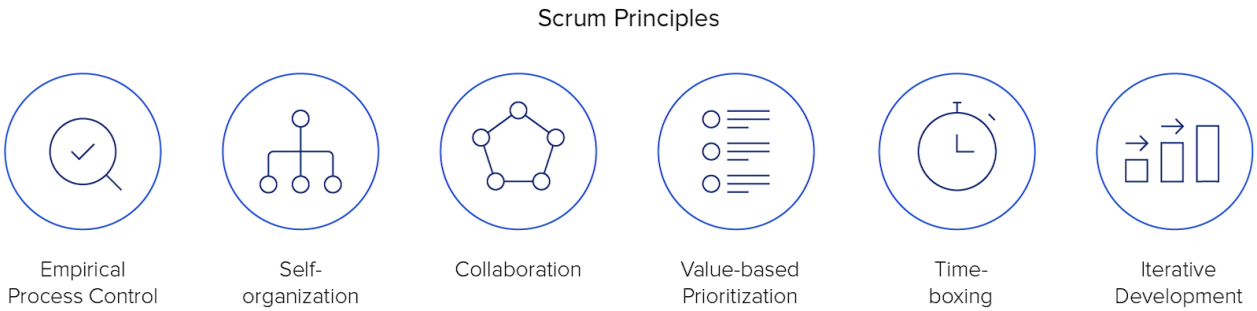
Scrum Values

According to the Scrum Alliance:

> Scrum is a simple yet incredibly powerful set of principles and practices that help teams deliver products in short cycles, enabling fast feedback, continual improvement, and rapid adaptation to change.

Source: scrum.org

Scrum is a prescriptive, incremental and iterative framework for developing software that applies Agile principles. The Scrum values and principles are outlined in the charts below and have significant alignment with Lean and Agile values and principles.



Source: SQLI group

Scrum Overview

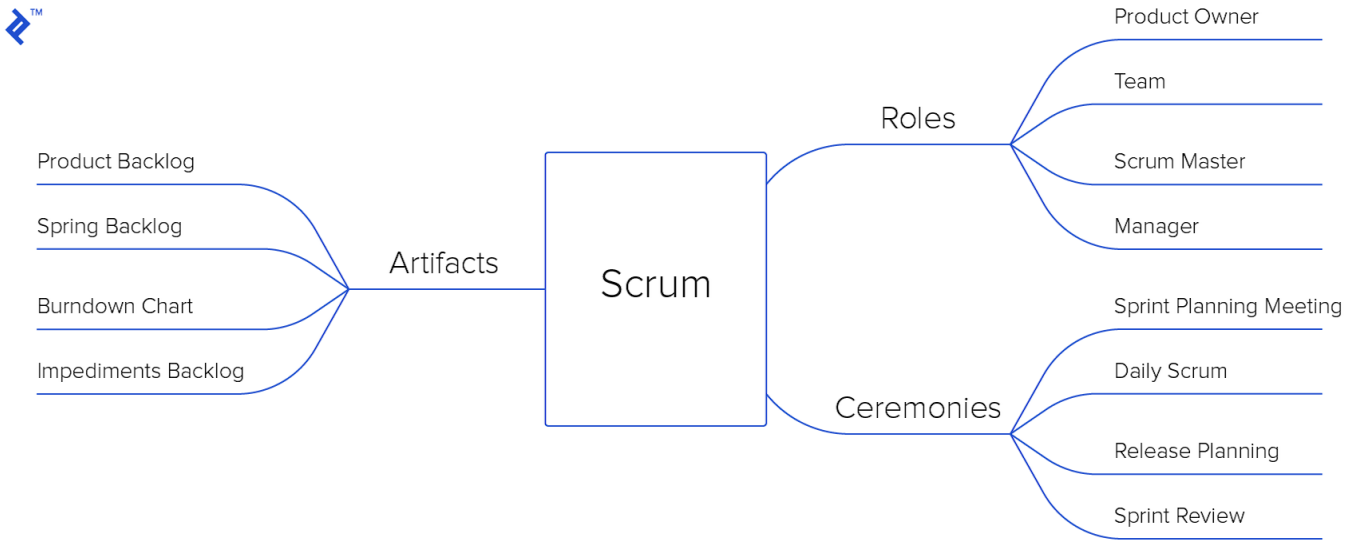
Work is divided up into short time-boxed iterations called sprints which are usually 1-3 weeks. This is in stark contrast to the in-depth planning of Waterfall. The work planned for the current sprint is chosen from the top of a prioritized backlog of work items called a Product Backlog (Pull System, Heijunka), and it is fixed once the sprint starts. The goal is to have working software at the end of each sprint, enabling fast feedback.

At the end of the sprint, the team meets to review the completed work done, how the sprint went, and to plan the next sprint. The sprint length, as well as the sprint rituals and cadence, are fixed for each sprint.

Sprints are executed by cross-functional teams containing all the skills needed to complete the work in the sprint. Daily planning and tracking of progress is done using visual artifacts like the scrum board and sprint burndown charts.

The work in a sprint is pulled from a prioritized backlog. Following these methods allow for changing requirements over time and encourages constant feedback from the end users.

The mind map diagram below outlines some of the main concepts of Scum which will be discussed in the upcoming sections.



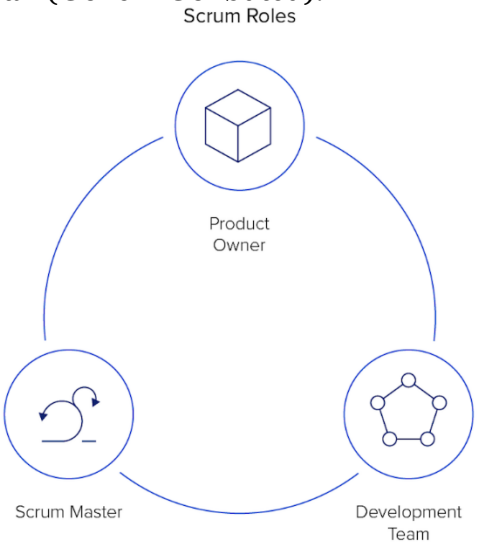
Scrum Roles

Scrum has three roles:

- **Scrum Master:** the Scrum Master is is a servant leader of the Scrum team. They are the coach of the team who helps facilitate collaboration, removes impediments, enforces and safeguards the Scrum process, and protects the team. This typically means that they organize and facilitate the sprint rituals, ensure that the product owner has a properly prioritized and groomed backlog, ensures that the team isn't pressured to over-commit to a sprint, ensures that scope isn't added to a sprint, ensures that the Definition of Done is adhered to. They do not assign tasks to team members (Genchi Genbutsu) and they are not responsible for the delivery of a project
- **Product Owner:** the product owner is 'the single wringable neck' responsible for delivery of the product. The product owner defines the vision of what they

want to build and conveys that vision to the team and the organization. The product owner owns the business and market requirements and prioritizes all the work that needs to be done thru creating and managing the product backlog. They decide which features to ship when. They work with the team and other stakeholders to make sure everyone understands the items in the product backlog. They accept or reject work completed in a sprint at the sprint demo.

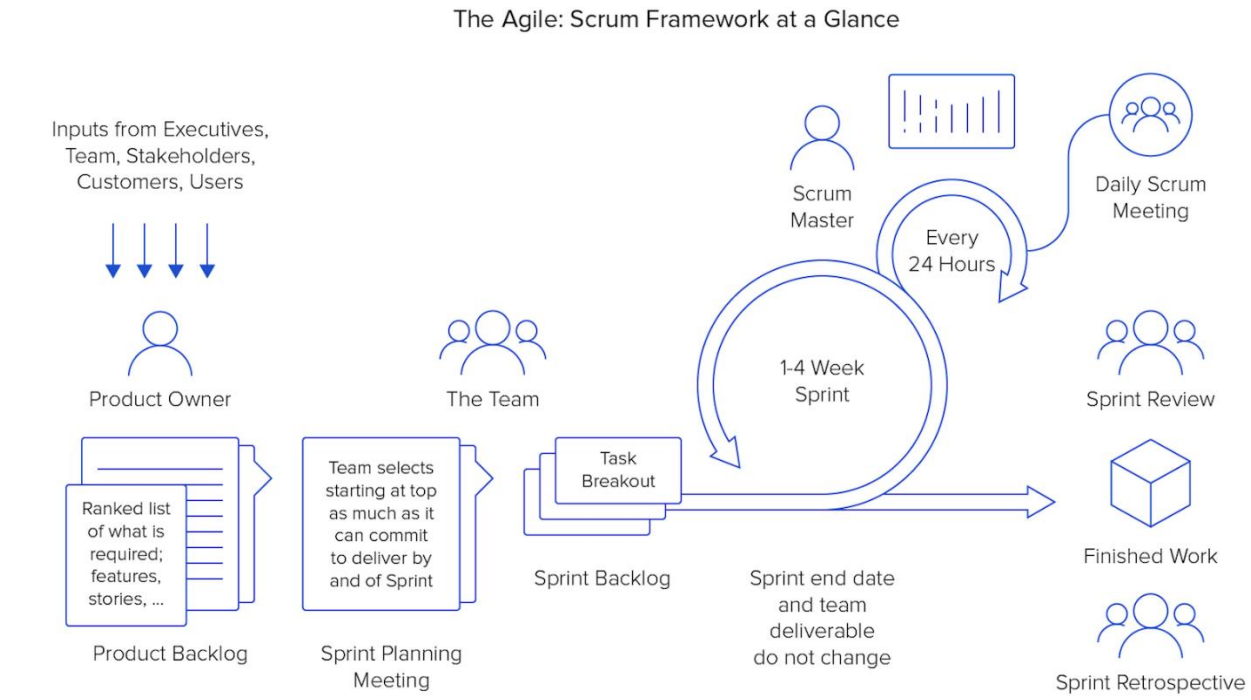
- **Team Member:** The Scrum team is a self-organizing, cross-functional team typically comprised of five to seven members. Everyone on the project works together and helps each other and not necessarily bound to distinct roles like architect, programmer, designer, or tester. Everyone completes the set of work together. The Scrum team plans how much work they can complete each sprint and owns the plan (Genchi Genbutsu).



Source: Scrum Alliance

Scrum Flow, Activities, and Ceremonies

As discussed above, Scrum has a defined flow and a set of rituals and activities. The flow of a sprint is as follows.



Source: Scrum Alliance

SPRINT PLANNING:

Before a sprint starts, the Scrum Master facilitates a meeting with the scrum team and the product owner, called the sprint planning meeting, where the product owner identifies the objectives of the upcoming sprint, and the team then plans their work according to the objectives.

This is usually done with the following activities:

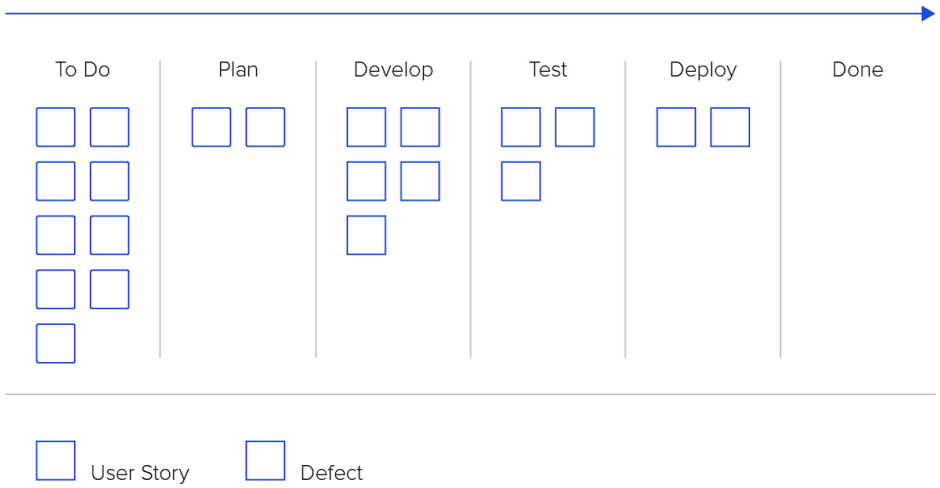
- **Sprint Capacity:** the team determines the capacity for the sprint, taking into account the number of days, number of team members, holidays, etc.
- **Sprint Objectives:** the product owner identifies what the objectives are for the sprint. It is critical that the product backlog is prioritized according to the objectives (i.e., the relevant stories are at the top) and groomed.

- **Work Selection:** stories or tasks are pulled from the top of the backlog into the sprint until the estimated capacity is reached. As stories are pulled in, the product owner will explain the story and answer questions from the team, updating the story as needed, until the product owner and the Scrum team have a good, common understanding of the story. Once this activity is completed, the team has a proposed initial sprint scope.
- **Task Breakdown:** the Scrum team discusses each story in detail with emphasis on planning out how they will complete the story and address all the acceptance criteria and the DoD. They will produce a list of subtasks needed to complete the story. Once the list of subtasks is complete, the estimate of the story is reviewed and updated if necessary.
- **Sprint Commitment:** once all the stories are broken down and estimates are updated, the proposed initial sprint scope is reviewed. Stories may be removed from the sprint and put back on the backlog and/or additional stories may be added. Once this is done, ONLY the Scrum team (and not the Scrum Master or product owner) commits to completing the work in the sprint, and the sprint is started.

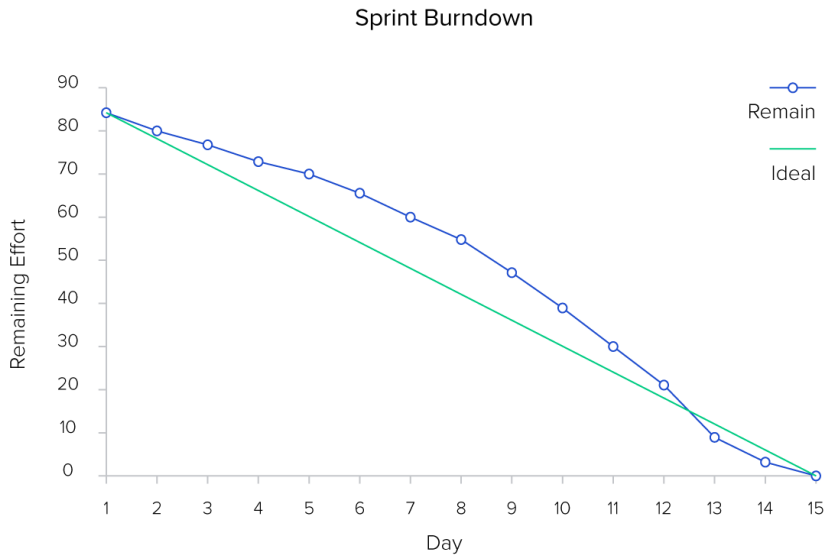
The total amount of work or scope committed to for the sprint is measured in story points.

SPRINT EXECUTION

During the sprint, team members pull work items (user stories, tasks, etc.) from the top of the sprint To Do list to work on. Various team members will work on the various work items or their subtasks. They will update the status of an item when appropriate by moving it from one column to the next (typically **To Do > In Progress > Testing > Done** or some variation thereof) on the Scrum Board until it is done.



Progress is tracked using a burndown chart which shows the amount of work completed and remaining measured in story points. Remaining story points are shown on the Y axis and the remaining time is shown on the X axis. The burndown chart is updated when stories are done.



Source: Scrum Alliance

On a daily basis, the Scrum Master focuses on:

- Facilitates the daily standup meeting to plan the day and review progress (see below)

- Ensures that the team has a plan for the day
- Removes roadblocks
- Protects the sprint scope and the team from distractions
- Helps the team maintain their burndown chart and other Scrum statistics

Daily Standup

At the beginning of each day in the sprint, the Scrum Master facilitates a brief, 15-minute meeting with the scrum team and product owner to plan out the day and review sprint progress. This is a short meeting where everyone is standing in front of the Scrum Board and each person in the meeting answers the following questions in 2 minutes or less, referring to specific items on the sprint board:

- What did you do yesterday?
- What are you planning to do today?
- Are there any impediments preventing you from completing your work?

This allows everyone to see what everyone is working on, see what progress is being made or not made, and identify impediments and/or help needed.

SPRINT COMPLETION

The Scrum Master facilitates two ceremonies to close out of the sprint before planning the next sprint.

Sprint Demo

At the end of the sprint, the Scrum Master facilitates a sprint demo meeting where each completed story is demonstrated on the working software to the product owner and the rest of the team. The product owner will either accept the story if all the acceptance criteria are met, or they will reject the story. If a story is rejected, the shortfalls are identified and the story is put back onto the product backlog in its priority order to be completed at a later time or not at all. Often, the parts of a story that the product owner doesn't accept are broken off into a separate story(s) and the original story is closed.

The total number of completed story points per sprint (Velocity) is calculated and the sprint is closed. The velocity is used to track the team's output level and is used to estimate when features or releases will be complete.

Sprint Retrospective

After the sprint demo but before the next sprint planning meeting, the Scrum Master facilitates a sprint retrospective where the team reflects on the sprint that just completed and discusses what went well and what didn't go well so that they can continuously and incrementally improve process and quality over time (Kaizen, Hansei). There are a plethora of retrospective formats or exercises that can be used to help the team generate discussion.

A list of improvement action items is produced and sometimes they result in items being added to the product backlog, changes to the DoD or team charter, etc.

PRODUCT BACKLOG MANAGEMENT

Product Backlog Creation

Before a sprint can be planned or executed, the product owner needs to create a product backlog of work. The backlog usually starts with feature development items called

stories written by the product owner and over time also becomes populated with development or QA tasks, spikes, and defects, etc. potentially created by any member of the team. All the items in the backlog are arranged in priority order.

Backlog Grooming

Once the initial product backlog is created and prioritized, the ongoing backlog groom process takes over. The goal is to always have, at a minimum, enough of the top items in the backlog groomed and estimated so that they are ready to be pulled into a sprint during a sprint planning meeting. This is typically done by having regular ongoing backlog grooming meetings with the product manager and the team facilitated by the Scrum Master.

Prior to the meeting, a list of stories is sent to the team so they can review and prepare for the grooming meeting. At the grooming meeting, each item is discussed in terms of the acceptance criteria, complexity, risk, size, implementation strategy, etc. The acceptance criteria and other details of the story are reviewed and revised until the team, product owner, and Scrum Master have a common understanding of the story. At that point, the story is estimated in story points using a technique called planning poker.

Story Point Estimating

Story points are a unit of effort that uses relative sizing where stories are compared against previous, known, well-understood pieces of work. You are always asking the question “is this story bigger, smaller, or approximately the same size” as some other piece of work.

The Fibonacci scale (1, 2, 3, 5, 8, 13, 21…) is the most commonly used scale where each increment is approximately twice as big as the previous (i.e., a five-point story is more or less twice as big as a three-point story). Sometimes other scales like T-shirt sizes (XS, S,

M, L, XL) or even fish (minnow, goldfish, trout, tuna, whale, etc.) are used. Any scale that allows you to compare the size of something relative to another will work.

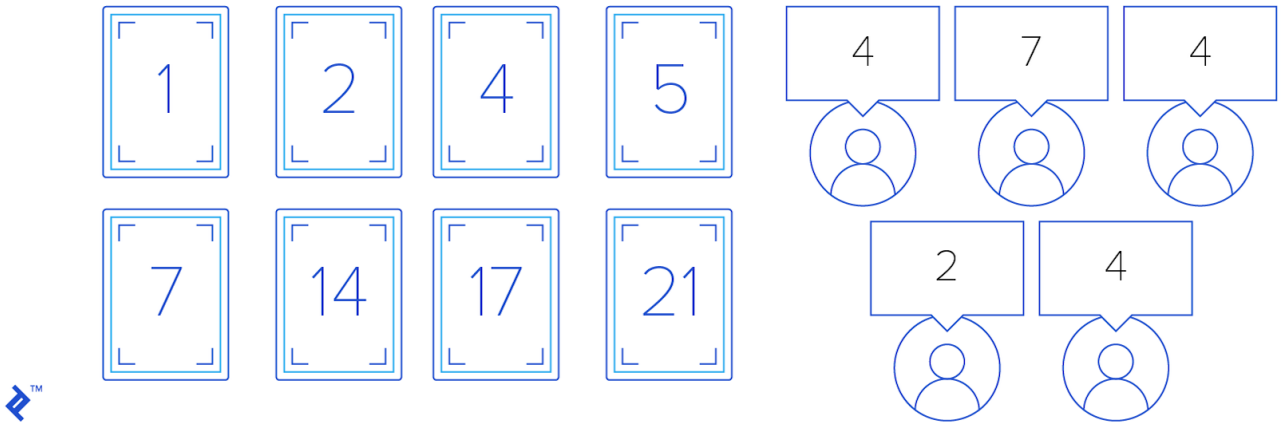
Story points represent the team's entire effort to implement a story, including development, testing, design, and other miscellaneous tasks needed to the Definition of Done. The estimate takes into account the amount of work, complexity, and risk. Once the relative size of the story is determined, a size on the scale is assigned as the estimate.

Teams prepare for story point estimating process by first establishing a baseline for estimation by picking a "most medium" sized story that the whole team understands as a reference story. Some additional reference stories that are bigger and smaller are also chosen.

Story point estimating is done during grooming meetings and sometimes during the sprint planning using Planning Poker:

1. Each team member/estimator has a set of cards.
2. Backlog items are discussed one at a time as described above.
3. Once the item has been fully discussed, each estimator privately chooses a card to represent their estimate.
4. When all estimators have made their estimates, they reveal their cards at the same time.
    - If all estimates match, estimators select another backlog item and repeat the same process.

- When estimates differ, the estimators discuss the issue to come to a consensus.



The advantages of story point estimating are:

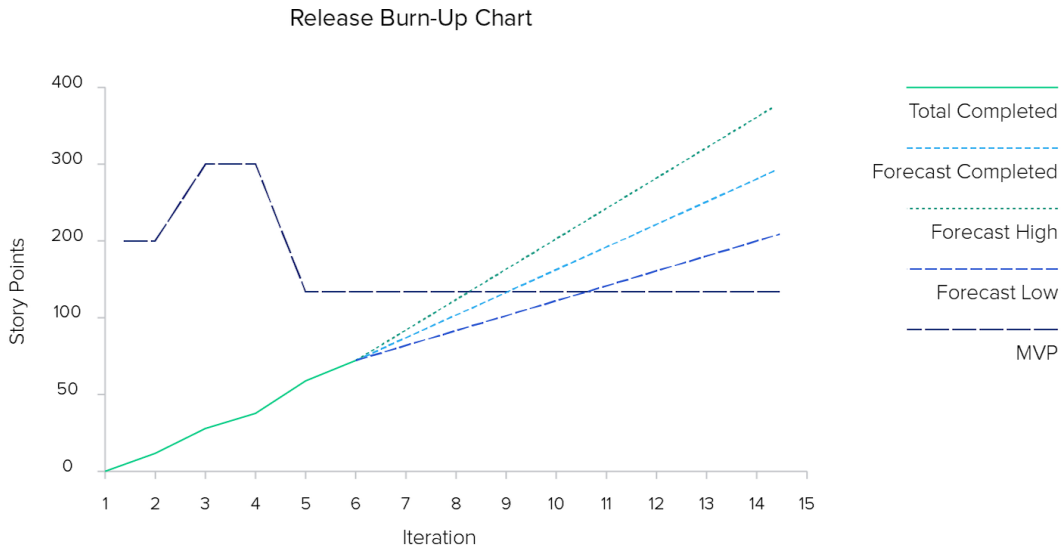
- **Quick:** estimates are relative to already completed product backlog items.
- **Accurate Enough:** accurate enough to give an overview of the scope, plan future work, prioritize, and manage expectations.
- **Embraces Uncertainty:** Story points specify an unknown time range. Selecting from a specific Fibonacci-like sequence of story points allows capturing uncertainty.
- **Team Sport:** Involves everyone (developers, designers, QA, product managers). Uses multiple perspectives to determine the size of work but only team members doing the work can estimate
- **Measures Team Velocity:** measures how much work a team does in a sprint versus the amount of time spent doing the work. As the team improves, they will complete stories of the same size quicker, resulting in a higher story point velocity over time.

Release Estimating and Tracking

Story point estimating is also used in a release planning context using the following technique:

1. List all the stories to be sized
2. Put them in order from smallest to largest
    - Take the first and second user story.
    - Decide which is bigger and put the bigger one below
    - Then take the next one and decide where it fits relatively to the other two
    - Repeat the process until all the stories are now in the list (in a sequence from smallest to largest)
3. Size the stories

The story estimates for all the stories in a release combined with the team's velocity will allow you to estimate when a release will be complete and track its progress. This is often shown in a release burn-up chart.



Source: Scrum Alliance

The Scrum Artifacts and Terms

- **Product Backlog:** A backlog list of all work items for a given product which can include features (stories), technical tasks, spikes, and defects
- **Release Burn-up:** A graphical chart used to show progress at a release level and to predict when a release will be finished using Sprint Velocity. Completed story points are shown on the Y axis and sprints are shown on the X axis.
- **Sprint Backlog:** A backlog list of all work items to be completed in a given sprint. The contents of the sprint backlog are agreed on during the sprint planning meeting.
- **Scrum Board:** A table style board that tracks the progress of the work committed for the sprint. Statuses are shown on the top in vertical columns and work items are moved across each status until they are done. The scrum board is populated during the sprint planning meeting and is reset at the end of a sprint.
- **Sprint Burndown:** A graphical chart which shows the amount of work completed and remaining measured in story points over the length of the sprint. Remaining story points are shown on the Y axis and the remaining time is shown on the X axis.
- **Sprint Velocity:** The number of story points that a Scrum team completes in a sprint.
- **Impediments Backlog:** A list of impediments that need to be addressed by the Scrum Master so that the team can continue to work. When a team member is blocked, they will add an item to the impediments backlog to provide visibility to the team and Scrum Master.
- **Epic:** An epic is a large body of work consisting of multiple related user stories.
- **User Story:** A user story is a description of a software feature from an end-user perspective. The user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement and includes acceptance criteria. User stories are created and maintained by the product owner.

- **Task:** A task is a piece of work that is entered by the Scrum Master or team member that may be directly or indirectly related to user stories. They are usually technical in nature and will include acceptance criteria.
- **Spike:** A spike is a special type of task that is used when you need to research, prototype, and/or architect sometime before you can decide on how to implement or estimate a user story.
- **Subtask:** A subtask is a task that is entered as an implementation step towards completing a user story or task. They are usually entered by the team during a sprint planning meeting.
- **Story Point Estimates:** A relative sizing estimation scale that is based on the Fibonacci scale (1, 2, 3, 5, 8, 13, 21…)
- **Acceptance Criteria:** The list of story specific, testable items included in every story that must be completed before a product owner will accept a story as being completed.
- **Definition of Done (DoD):** A list of common steps or criteria that must be completed before any story can be considered done. The team agrees on this and documents it so that it doesn't have to be listed in every story.

Advantages and Disadvantages of Scrum

The primary advantage of Scrum is the application of Agile values and principles as well as Lean concepts such as Seiri, Jidoka, Just-in-Time, Kaizen, Genchi Genbutsu, Heijunka, Pull System, and Iterations. Applying these principles allow project teams to receive continuous feedback, quickly adapt to changing requirements and uncertainty, reduce waste, increase visibility and transparency, and strive for continuous improvement. By always focusing on the most important items in the product backlog and only working in short iterations that always produce working software, Scrum is more customer focused and allows customers to see what they like (and don't like) and

make changes as needed. The overhead cost in terms of process and management is smaller, thus leading to quicker, cheaper results.

Scrum is a great methodology for projects with requirements that are not clearly known and/or expected to change. This applies to most projects. Scrum is also great for experienced, motivated teams as it allows teams to organize the work themselves and it provides visibility, transparency, and accountability for both progress and problems. All of this will result in teams improving and becoming more productive over time.

Scrum does have some disadvantages and is not the best methodology in some situations:

- **Transparency:** Scrum increases transparency and accountability which is both an advantage and disadvantage as problems and poor performance within and outside the team and are exposed. This can be uncomfortable and can lead to resistance if not handled properly within the Scrum framework of continuous improvement.
- **Team Experience and Commitment:** Inexperienced and/or uncommitted Scrum teams or Scrum Masters can cause serious problems through misapplying the Scrum methodology. There are no defined roles in the Scrum team as everyone does everything, so it requires committed team members with technical experience to follow the Scrum process and improve over time. It can also be very detrimental if other parts of the organization are resistant to Scrum.
- **Scope Creep:** There is a risk of scope creep, especially if there isn't a defined end date as stakeholders are allowed to add to the scope. Being able to change scope and priorities is one of the main advantages of Scrum, but it can also be a disadvantage if discipline isn't used.

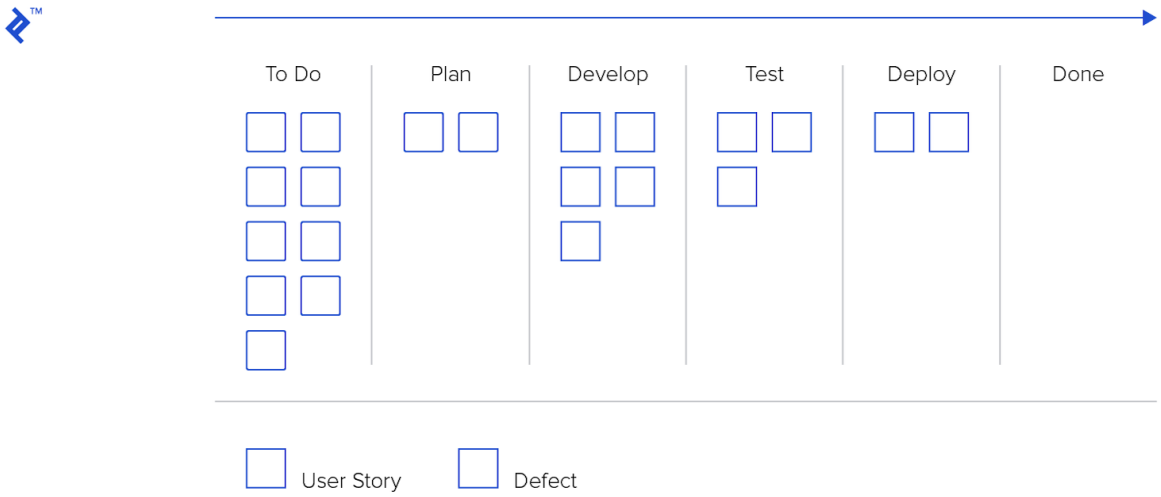
- **Poorly defined work:** Poorly defined and understood user stories or tasks can lead to rework, inaccurate estimates, and scope creep. Although Scrum is focused on working software over documentation, the product owner needs to be clear on what they want and be able to clearly communicate that in conversations and in the user stories.
- **Scaling:** Adopting the Scrum framework in large teams is challenging as Scrum is meant for smaller teams.

Kanban

What Is Kanban?

Kanban is a lighter weight process that applies many of the Lean and Agile values as well as a subset of the Scrum values and principles but there are also some fundamental differences. Kanban focuses on visualization, flow, and limiting work in progress.

Kanban is Japanese for “signboard” or “billboard.” Toyota line-workers used a kanban (i.e., an actual board) to signal extra capacity in various steps in their manufacturing process. In software, this is done with a Kanban board, which is very similar to the Scrum board. There is a prioritized backlog of To Do items and vertical columns for each status that a work item can have. Like Scrum, work items are moved from one status to another; however, in Kanban, the amount of work in progress is strictly limited to a maximum number of items in each status based on team capacity. New work cannot be pulled in until existing work is moved to the next step in the process. In Scrum, work in progress in indirectly limited by controlling the amount of work planned for a sprint.



In Kanban, there are no fixed iterations or sprints, just a constant flow where work items are pulled from one stage to the next. This means that the Kanban board is never reset. It also means that many of the Scrum rituals not used. Kanban teams often have daily standup meetings but they are not prescribed. There are no regularly planned sprint planning meetings, sprint demos or sprint retrospectives, so the process is more lightweight. Some of the activities in those rituals may or may not be performed at an informal level. Continuous improvement is accomplished by tracking and analyzing the flow of items from one state to another and making constant incremental improvements based on what issues are uncovered.

Kanban also doesn't prescribe the roles that Scrum does. The only role needed is the team member role, however, there is often a project manager that manages the team and the backlog. Often a single Kanban board can serve multiple function based roles and/or teams that only work on items in a certain status. For example, a development team might pull items from To Do to In Progress and move them to Testing, and the test team might test items in Testing and move them to Done.

Many of the backlog management activities for prioritizing and grooming of work items still need to be done to ensure that a given work item is well understood and ready to be worked on, however, estimating the work items is prescribed as optional.

Kanban vs. Scrum

The following table compares Scrum and Agile:

Kanban	Scrum
Continuous Delivery	Timeboxed Sprints
Less process and overhead	Has prescribed Sprint rituals and roles
Focuses on completing individual items quickly	Focuses on completing a batch of work quickly
Measures Cycle Time	Measures Sprint Velocity
Focuses on efficient flow	Focuses on predictability
Limits WIP for individual items	Limits WIP at a Sprint level
Individual work items are pulled	Work is pulled in batches at Sprint Planning
No prescribed roles	Has prescribed roles (Scrum Master, Product Owner, Team Member)
Kanban Board can be organized around a single cross-functional team or multiple specialized teams	Scrum Board is organized around a single cross-functional team
Changes can be made at any time -> more flexible	Changes are only allowed in the Product Backlog. Changes within a sprint are not allowed
Requires less training	Requires more training

Kanban	Scrum
Good for teams where only incremental improvements are needed	Good for teams where fundamental changes are needed

Summary: The End of Part 1

In this part, we reviewed a few of the most popular methodologies used for Software Development. By now you should have a good understanding of Lean, Agile, Scrum, and Kanban and their historical roots in Lean Manufacturing and TPS. In the next part of the series, we will continue reviewing and comparing other Software Development methodologies such as Waterfall, JTBD, and SAFe (and other scaling frameworks), as well as hybrid methodologies, so you have all of them conveniently explained in one place.

UNDERSTANDING THE BASICS

What is Agile?

Agile is an umbrella framework that applies to any process that applies the Agile set of values and principles. Some of the examples are: Extreme Programming, Scrum, and Kanban.

What is Scrum?

Scrum is a simple yet incredibly powerful set of principles and practices that help teams deliver products in short cycles. Scrum is a framework that applies Agile principles that was invented separately by multiple people, several of whom signed the Agile Manifesto.

What is Kanban?

TAGS

ProjectManagementAgileLeanScrumKanban

Darren Hagman
Agile Project Manager

ABOUT THE AUTHOR

Darren is a veteran developer, scrum master, and project manager with deep experience in both Waterfall and Agile methodologies from working in the transportation, archiving, eCommerce, and aerospace industries. Darren spent ten years working in developer and lead/architect roles before becoming a project manager. Darren understands the challenges of implementing Agile practices in a team and how to effectively interface with the Waterfall world.