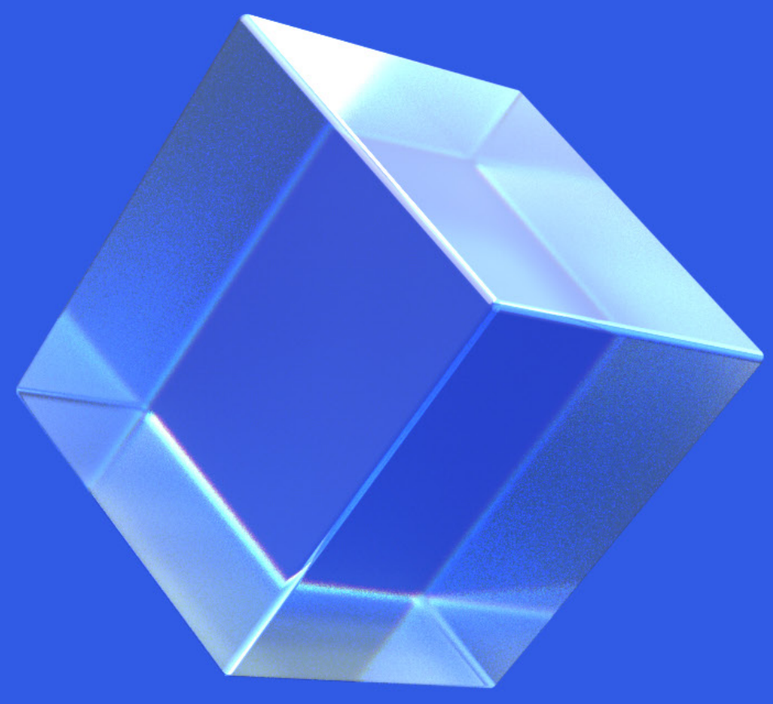
by **airfocus**



Agile:

Best Practices and Methodologies

Table of contents

PS: If you are in a rush and would like to try airfocus for free today then click here to get your 14-day trial version.

Why We Wrote This Guide

If you work within the software industry then you most likely work in an Agile environment or some combination of it.

However, do you know what Agile means, where it came from, and how it's employed? What about Scrum and Kanban? These are two words that you may have come across. Do you know how they relate to Agile?

Product managers work with their customers to understand their key pain points, prioritize them, and define the solutions to solve these pain points.

However, how do these solutions go from problem definition to tangible products that provide value for customers and users? This is done via Agile software development practices.

We wrote the Ultimate Guide to Product Management to shed light on what product management was and its evolving history, and also the Ultimate Guide to a Product Manager's Job to clarify the key responsibilities of those who carry the title "product manager".

We decided to write The Ultimate Guide to Agile to tie the loop and show how software development teams work with product managers to take a solution from idea to delivery.

While there are multiple online resources that speak about various Agile related topics, this detailed and comprehensive guide will save you the time of sifting through them and give you everything that you need to know.

Here are some of the main topics that this guide will cover:

- What is Agile
- Waterfall software development and how it works
- Scrum software development and how it works
- Kanban and how it works
- Pros and cons of Waterfall, Scrum, and Kanban
- Agile certification
- Agile tools
- And more

Let's get started.

Three Main Frameworks for Software Development

One misconception that many have when it comes to Agile is that they assume that it sole-ly concerns the development team.

Product managers also play a major role in Agile processes as they work closely with their teams to ensure that their key stakeholders achieve their intended value from the product and that they are releasing features and enhancements that assist their company towards reaching its goals.

It's important to note the three main software development frameworks before we dive deeper into Agile. These three frameworks are:

1. **Waterfall**
2. **Agile**
3. **Wagile**

   (a hybrid approach
   of both waterfall and agile)

To get a better understanding of Agile and its benefits, let's begin with what came before it: Waterfall.

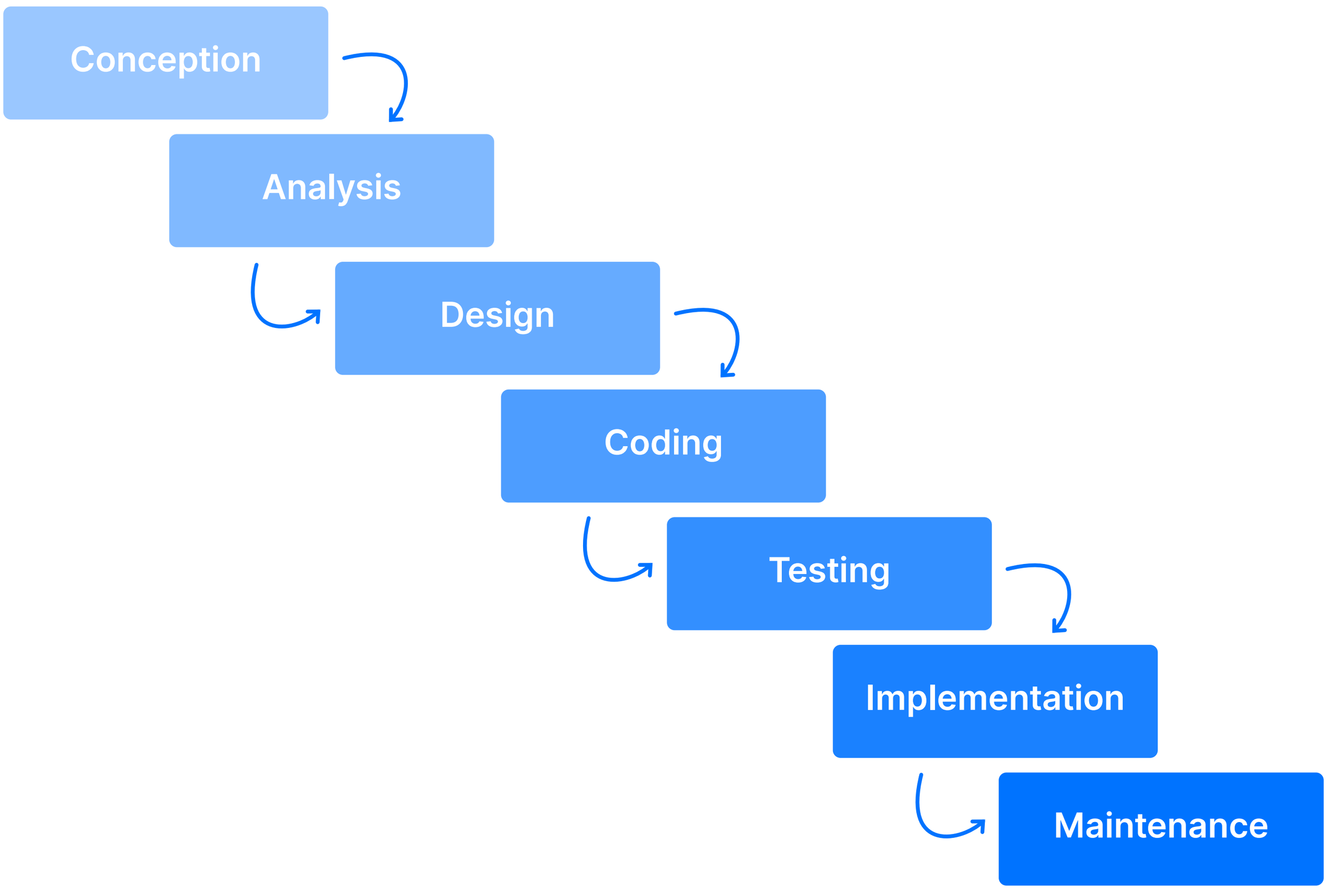
Waterfall

Waterfall was extremely popular pre-2000 and is still heavily used today. It is especially used for project management work and when working with agencies.

Its origins stem from software developers refining the stages of hardware development to produce software.

The stages of the Waterfall software development

Waterfall software development has 7 main stages according to the Software Development Life Cycle (SDLC):



With Waterfall software development a team does not move forward to the next stage until the current one is 100% complete.

100% complete means that all of the deliverables for that specific stage have been approved by the key stakeholders. Everything moves from one stage to the next like a big dump of water. Just like a waterfall.

Now you understand why it's called "Waterfall" software development.

Benefits of Waterfall software development

Though **presently it is not the most popular framework** to use when building software, Waterfall has many benefits.

Some of the benefits of Waterfall are:

1

Crystal clear specifications

With Waterfall stakeholders are crystal clear on what the final solution will be.

This is due to the detailed specifications that are defined in the early stages of the project, much prior to a single line of code being written.

2

More optimal design decisions

Due to an early fixed agreed upon scope designers and developers can make more optimal design decisions rather than designing for unknowns.

3

Works great with third-parties

Waterfall is great when working with agencies and/or third parties. Due to the stage-by-stage approach they can scope and budget based on agreed upon work.

This also makes it easier to manage a budget.

4

Frees up additional time for product managers

One benefit that Waterfall has for product managers specifically is that they are not as involved in the day-to-day activities related to delivery.

Once the requirements are set the team gains clarity on what their tasks are and a product manager can attend to other matters as they are needed.

5

Clear expectations from the C-Suite

The benefit that Waterfall poses to management teams is that it gives them clear specifications on what will be delivered according to the set milestones.

This makes it easier for them to strategize and plan accordingly. Especially if there are dependencies related to the work.

Downsides of Waterfall software development

For the many benefits of Waterfall software development there are also downsides as well.

What are the downsides of Waterfall?

1

Not suited to handle constant changes

Waterfall is not the best software development framework to follow when there are constant changes to the market, product, stakeholders, etc.

Once the requirements have been finalized there may be new information that is received that drastically affects the product. Unlike Agile, Waterfall does not make it easy to go back and change decisions made in the earlier signed-off stages.

Which leads to downside #2.

2

Good decisions may turn out to be incorrect, and it’s much too late to change

With any product there are assumptions that are made along the way.

One way to validate your assumptions is by placing your product into the hands of your customers for direct feedback.

With Waterfall however the time may be lengthy to complete each stage and gain approval from stakeholders before moving forward.

While business leaders may assume that customers wanted a specific feature with a specific implementation, with the time required to get through all of the stages and launch, once the product has launched market sentiments may have changed.

Changes at this point may be costly and time consuming.

If they are critical then the team may now have to go back and revisit their flow, then go stage-by-stage before launch.

3

Potentially late releases

While this can happen with Agile as well, one thing to note is that with Agile there are incremental releases for the product. So even if a product is late, customers will still have a real working product that they can benefit from soon.

Anyone who works in the software industry knows that software development takes longer than planned. There are many unknowns that may occur.

For this reason product managers try to factor these unknowns into their plans.

Though there might be a clear scope with set milestones, if other stakeholders are dependent on the plan and something slips, then future stages get delayed and it takes a longer time for customers to get access to a working product.

4

Testing is saved until the later stages

With Waterfall, testing does not happen until stage 5: Testing.

This can be problematic because there may be unnoticed issues in the product which are not discovered until later stages. And these may be critical issues.

There is a lot of work that goes into the stages prior to step 5, work which has been finalized and signed off on to progress to the next stage.

Once an issue is found after testing the team will be required to go back and address these issues which could have been discovered prior had testing taken place prior. Or in tandem with development.

Agile is a direct response to this.

5

Customers aren't included in the product development process until launch

Adding to the above, with Waterfall customers and end users must wait until the end of Implementation (stage 6) before they can obtain value from it.

The challenge this poses is that their sentiments may have changed from the time the team started discovery to the launch of the product.

And if there are issues post launch or the product no longer meets their expectations then it is more costly and time consuming for the team to address them.

When to use Waterfall

Keeping the benefits and downsides of Waterfall software development in mind, when is it best to use it?

Waterfall should be used when the requirements of the work to be performed are crystal clear.

For example when building computer operating systems or critical software applications.

It would not make sense to build car braking software in an Agile way; build a portion of it, launch, test, and then iterate. People's lives are too important to release such critical software in such a fashion.

Rather a solution as important as this needs to be thoroughly thought out from beginning to end, rigorously tested, and then launched.

If the requirements are clear and there is little expectation of change to the scope then Waterfall is the way to go.

At this point you're probably asking yourself: if this is an ultimate guide about Agile why did it start by covering Waterfall?

Knowing what came before Agile will give you a better appreciation for it and a better understanding of how and why it works.

Let's now dive deeper into the main topic of this ultimate guide.

What is Agile Approach?

While Waterfall is a linear, rigid, and sequential process for developing software, Agile takes a more flexible and iterative approach.

With Agile a team works within set periods of time (sprints or cycles) to deliver solutions incrementally which are meant to improve products while keeping the customer's needs at the forefront. With the customer's current needs at the forefront improvements are made to the **product based on feedback from customers post-release.**

While Agile is characterized by flexibility, embracing change, and making quick changes where necessary, **the most important characteristic of Agile is that it is focused on the customer and their needs.**

The origins of Agile

On February 12, 2001 a group of 17 individuals met at a ski resort in Snowbird, Utah to discuss and find common ground. The purpose of this meeting was simple: to discover a better way to build software.

These attendees of this meeting were representatives from Pragmatic Programming, Extreme Programming, Scrum, Adaptive Software Development, and more.

What was the result of their numerous discussions? **The Manifesto for Agile Software Development.**

This manifesto is a set of principles that outlines the Agile values. It states:

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

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

Check out the manifesto here. The original website is still available for viewing.

Agile Manifesto We Value:

Individuals & Interactions	over	Processes & Tools
Working Software	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding To Change	over	Following a Plan

Let's dissect these values a bit further:

1

Individuals and interactions over processes and tools

Individuals are the ones who respond to business needs and drive the development process.

When processes and tools drive development this leads to teams being less responsive to change. As a result you will not be able to meet user needs.

Understanding user needs comes from interacting with individuals, not from obsessing over processes and tools.

2

Working software over comprehensive documentation

An inordinate amount of time is spent on documentation for delivery, specifically the Product Requirement Document (PRD) that is heavily relied upon in Waterfall.

A PRD outlines all of the features and capabilities of a product to ensure that it is ready for release. If a feature is listed in the PRD but is not in the product then the product does not get released.

The extensive list of required documentation causes long delays in development, delays that can be costly given that market conditions can and do change.

Agile doesn't seek to remove documentation entirely, documentation is important. With Agile leaner documentation is produced, documentation which outlines the problem statements, user stories, and technical requirements for the team.

This provides the team with the information that they need to get started on their work without being bogged down by an inordinate amount of details.

Documentation is valuable however getting started with working software has greater value.

3

Customer collaboration over contract negotiation

Product managers should engage their customers and collaborate with them on a regular basis. This means including them as part of the product development process.

Doing this makes it easier for a team to validate their assumptions and gain product evangelists upon launch.

With Agile, customers are included in the product development process at regular intervals and shown work progressively for validation.

4

Responding to change over following a plan

Due to the shortness of iterations Agile makes it easier to shift priorities when needed from iteration to iteration.

Bugs can be fixed, enhancements can be introduced, and new features can be added with each iteration.

Changes always improve a product as they provide additional value. **It is the product manager's job to define and measure the value that these changes will bring.**

The principles of Agile

Agile has 12 principles that are derived from its four values. These principles are:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Business people and developers must work together daily throughout the project.

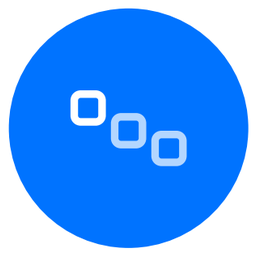
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

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.



Simplicity – the art of maximizing the amount of work not done – is e ssential.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

The best architectures, requirements, and designs emerge from self-organizing teams.

Continuous attention to technical excellence and good design enhances agility.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Benefits of Agile software development

Now that we have a better understanding of Agile and how it differs from Waterfall let's outline its many benefits.

Reduced risk

Due to releasing an initial MVP to the market that focuses on the most valuable feedback, obtaining quick feedback, and having incremental sprints, **following Agile reduces the chance of absolute failure.**

This is when teams aim to begin with a working product from the first sprint.

In the case where there are failures then the team can fail fast and make corrections as needed based on reliable data from customers who have received and tested the product.

Increased customer satisfaction

With faster releases customers are able to receive the initial MVP as well as added improvements sooner with each incremental release.

With frequent sprint reviews the team can demonstrate working functionality to their customers and key stakeholders can remain engaged throughout the product development process.

More predictable cost and schedule

With Agile, each work period (sprint or cycle) has a fixed duration. This leads to a more predictable cost that is limited to the amount of work that can be performed by the team per sprint.

For those who work within an agency for example, prior to each sprint clients will know the approximate cost of the work to be performed within that specific sprint.

This also helps improve the decision making for the priority of solutions and the need for additional iterations.

Improved product quality

Daily testing in the development process **allows the development team to address issues early** on.

The small incremental releases in Agile builds on previously tested functionality. Each release gets tested and validated for continuous software quality.

Along with this the frequent retrospectives enables teams to get better at their work. This is so long as they take the feedback from these meetings seriously.

Additional project control

There's transparency within the team and key stakeholders on exactly how work is progressing, the challenges present, and what's next. In addition, the regular sprint meetings (Scrum) make it easier for the team to assess where they are and what's next.

Project managing Agile teams can be a tough task however with tools like airfocus project leads can supercharge their team's agility and enable better focus.

airfocus combines the best elements of standard Agile software with some genuinely cutting edge innovations — making it one of the best Agile project management tools around.

Prioritize instantly with a number of readily available Agile roadmap templates, include team members in the decision making process via the Priority Poker function for collective prioritization, and among many other features, visualize project priorities with Kanban boarding.

Check out our list of The Best Agile Project Management Software in 2020 to learn more.

Relevant metrics

The metrics that are utilized in Agile software development **to estimate time, cost and measure performance** for data-informed decisions are more relevant and accurate than metrics for traditional projects.

With Agile you can determine the team capacity based on the team's actual performance (based on how the team performed in past cycles/sprints) and accurately predict what to expect next.

Likewise, as the work is implemented you can refine the estimated effort, time, and cost on a recurring basis as the team learns more and gains further familiarity with their work.

Higher team morale

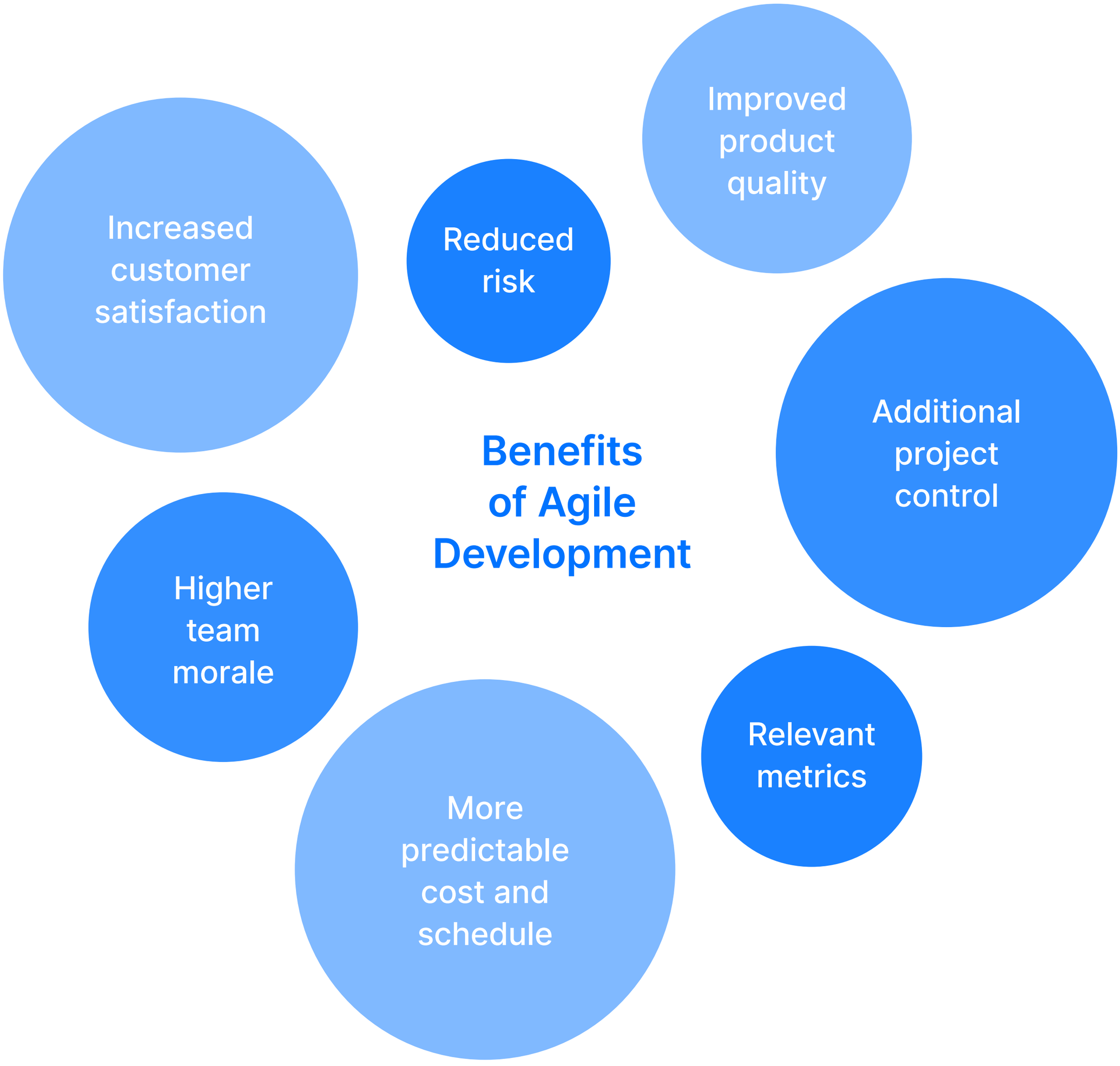
Being a member of a self-managing team allows individuals to be creative, innovative, and acknowledged for their expertise.

Also, unlike the sequential and siloed approach to Waterfall software development, having cross-functional work facilitates team members to learn new skills and grow by teaching others.

Everyone works together, not in silos.

One final thing to note is that **shipping a working product to customers and immediately receiving their positive validation boosts the team's morale.**

What more does a team want than to see that their hard work is benefiting their end users?



Downsides of Agile

With all of these listed benefits, what are some of the downsides of Agile?

Can lead to reactionary development

One of the claims that people have regarding the downsides for Agile is that it leads to reactionary development.

There is a semblance of truth to this.

This is because a team may build what users want now rather than planning a roadmap and maintaining a product vision for what they may need in the future.

However, if this **is the case it's not the fault of Agile but rather the product manager.**

It is their responsibility to manage the roadmap and ensure that customers and users are receiving value from the product in the present but will also continue to receive value from the product in the future.

One thing to keep in mind is that Agile is output focused, not discovery focused. Dual-Track Agile is what enables product teams to add a discovery track and focus on understanding problems first.

Busier product managers

When following Agile software development product managers are kept busy with the many tasks that they need to perform throughout their week.

Product managers already have a lot to do on a weekly basis. Their responsibilities include, but aren't limited to, gathering and prioritizing customer feedback, defining problem statements and hypotheses for the problems they want to solve, enabling their stakeholders for product launches, and maintaining the product roadmap (check out The Ultimate Guide to a Product Manager's Job.)

It's an added responsibility for product managers to manage their teams as they discover, design, develop, launch, and iterate on feedback for each release of their product(s).

Why Agile benefits product managers

What benefit does Agile provide to product managers specifically?

Well, for one, it enables them to get a product to market faster and gain insights from their customers.

The way that a product is improved post-launch will mainly be based on customer feedback and metrics.

With Agile a working product is shipped and then improved via incremental releases.

Once the product has been launched, a product manager will meet with their customers and users to get feedback, assess their metrics, and prioritize what to deliver next to continue with improvements.

Irrespective of the software development framework employed for implementing Agile product managers are still responsible for:

- Crafting and maintaining the product roadmap
- Communicating feature prioritization and functional requirements to their team
- Answering any questions their team has as work is implemented
- Removing any blockers and ensuring clarity throughout design and development
- Defining “done” and ensuring that the acceptance criteria are met prior to release
- Communicating with customers and gathering their feedback to improve the solution post-launch

The responsibilities of product managers remain the same whether Agile is employed or not.

The Hybrid Approach

Many companies do not follow pure Agile, rather they follow a blend of both Waterfall and Agile together to fit their needs.

There are multiple names for this mixed approach. Some of them include: Hybrid, Wagile, and Waterscrum.

With a Hybrid approach, the initial stages are spent on discovery, design, and development. And once the initial version has been developed then Agile is fully implemented.

Hybrid provides an iterative approach to test, launch, measure, and release software. And then post-release the cycle continues.

This is the common practice however teams do modify this approach to fit their needs.

Now that we have a good understanding of Waterfall vs. Agile, let's now discuss two of the most common Agile frameworks that are employed in software development today: **Scrum and Kanban.**

Scrum	Kanban	Scrumban
✗ Continuous flow	✓ Continuous flow	✓ Continuous flow
✓ Team Roles	✗ Team Roles	✗ Team Roles
✗ Easy To Adapt	✓ Easy To Adapt	✗ Easy To Adapt
✗ Reactive to Changes	✓ Reactive to Changes	✓ Reactive to Changes

One important thing to note regarding both is that while some companies employ one or the other it doesn't have to be this way. **Scrum and Kanban are frameworks used to implement Agile software development.** That being said, teams are free to, and do, blend elements of Scrum and Kanban to meet their specific project needs.

So what do you call Scrum mixed with Kanban? Scrumban.

Remember, Scrum and Kanban are **guides** to help develop software in an Agile way.

Scrum Framework

Let’s start with the most popular of the two.

One interesting thing to note regarding Scrum is that it was developed in the early 90s and some of the initial founders of Scrum were part of the group that created the Agile manifesto.

The key principle regarding Scrum is that teams functioning as one unit is a key factor of success.

When using Scrum as a framework for software development the team comes together to prioritize the required work and sets short term goals for the work that needs to be completed.

Roles and responsibilities in a Scrum team
There are 3 main roles in a Scrum team with each role carrying specific responsibilities.

Product owner

The product owner owns and manages the development backlog.

They ensure that it is prioritized **with the most important items** at the top and the least important at the very bottom. And any items which will not be addressed are removed.

They also write the specifications for each item in the format of user stories and acceptance criteria.

User stories clarify the goal of the intended work and the value that the end user will receive.

Acceptance criteria are written to clarify when the specific piece of work meets the definition of done.

Check out our guide on the 9 Common User Story Mistakes Most Product Managers Make.

The product owner is generally used as the term for the product manager in a Scrum team. They make the decisions on which items will move from the development backlog to the sprint backlog to be worked on in the sprint.

Scrum master

A scrum master's main responsibility is to facilitate the Scrum processes.

They ensure that everyone understands how things work, what Scrum means, and ensures that the processes are being followed. Not just within the Scrum team, but with the entire company.

Scrum team

The Scrum team consists of all of the other members of the team who work on the tickets that come from the backlog.

This includes the designers, developers, and the Quality Assurance team.

Scrum's 5 rituals

Scrum has 5 main rituals that are followed each sprint to facilitate the Scrum process.

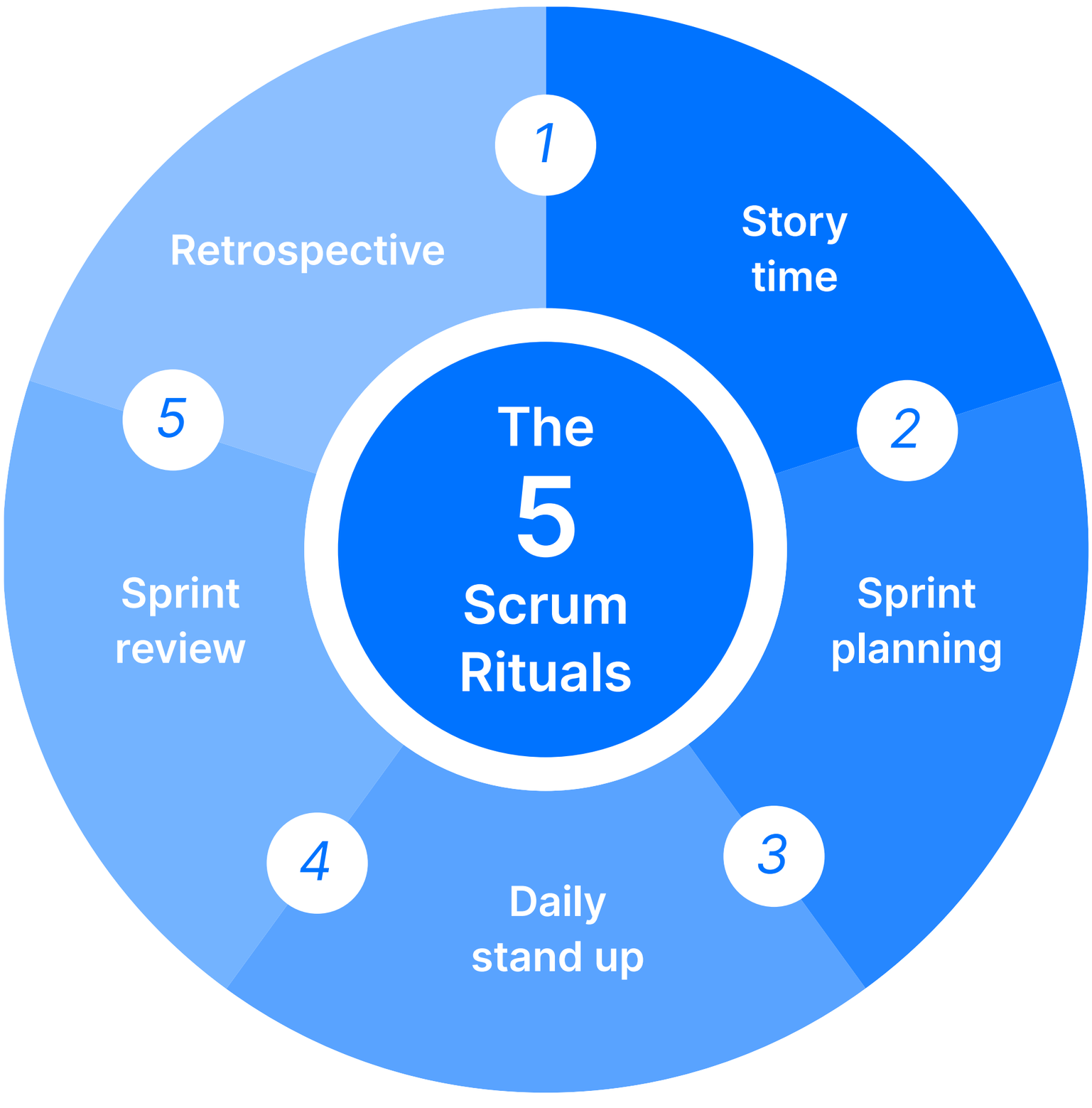
You can also think of these as 5 main events that take place every sprint.

A sprint is a period of time in which the team works towards accomplishing a specific goal. It is from when the work is planned to the point that the stated work is completed and released.

Sprints vary in length depending on the company.

While companies have 2 week sprints, sprints can vary from 2 to 6 weeks depending on the company.

The Scrum team gets together at the beginning of each sprint to plan the work for that sprint and the expectation is that at the end of the sprint there will be additional improvements that have been tested and made ready for release for the benefit of users.



The development backlog

Before we dive deeper into each Scrum ritual we want to shed some light on the development backlog. This is where “stories” come from.

The development backlog is a centralized list of prioritized tickets that include stories, tasks, spikes, bugs, and technical debt.

- Stories: New features or enhancements to be added to the product written in a specific format

- Tasks: Things that need to be taken care of by someone on the team. For example updating the copy on a specific page

- Spikes: Items that someone on the team needs to further research or investigate to inform future decisions

- Bugs: Issues within the product that needs to be fixed

- Technical debt: Rework that needs to be done for the product due to an intentional past decision

The development backlog is prioritized and maintained by the product owner.

Good product owners regularly review their development backlog and nurture it to ensure that the highest priority items are the most visible, near the top, while those lower in priority are the least visible, near the bottom.

DEEP backlog

If you want to have an effective backlog then follow the DEEP model. This is an acronym that clarifies how a development backlog should be managed.

Detailed: Each item contains the required details and has enough details for members of the Scrum team to work on them independently

Emergent: The backlog constantly evolves. Items are added as needed and likewise removed as needed. The work is flexible and can evolve based on the team’s needs, business needs, and/or customer needs

Estimated: Upcoming items should have estimations tied to them so that the team can accurately plan their work for the sprint(s). And those items with a high level of estimate are broken down into smaller pieces of work

Prioritized: Backlog items are ranked with the most important at the top and the least important at the bottom. This way as work for the sprint is complete then the next highest priority item(s) can be pulled in

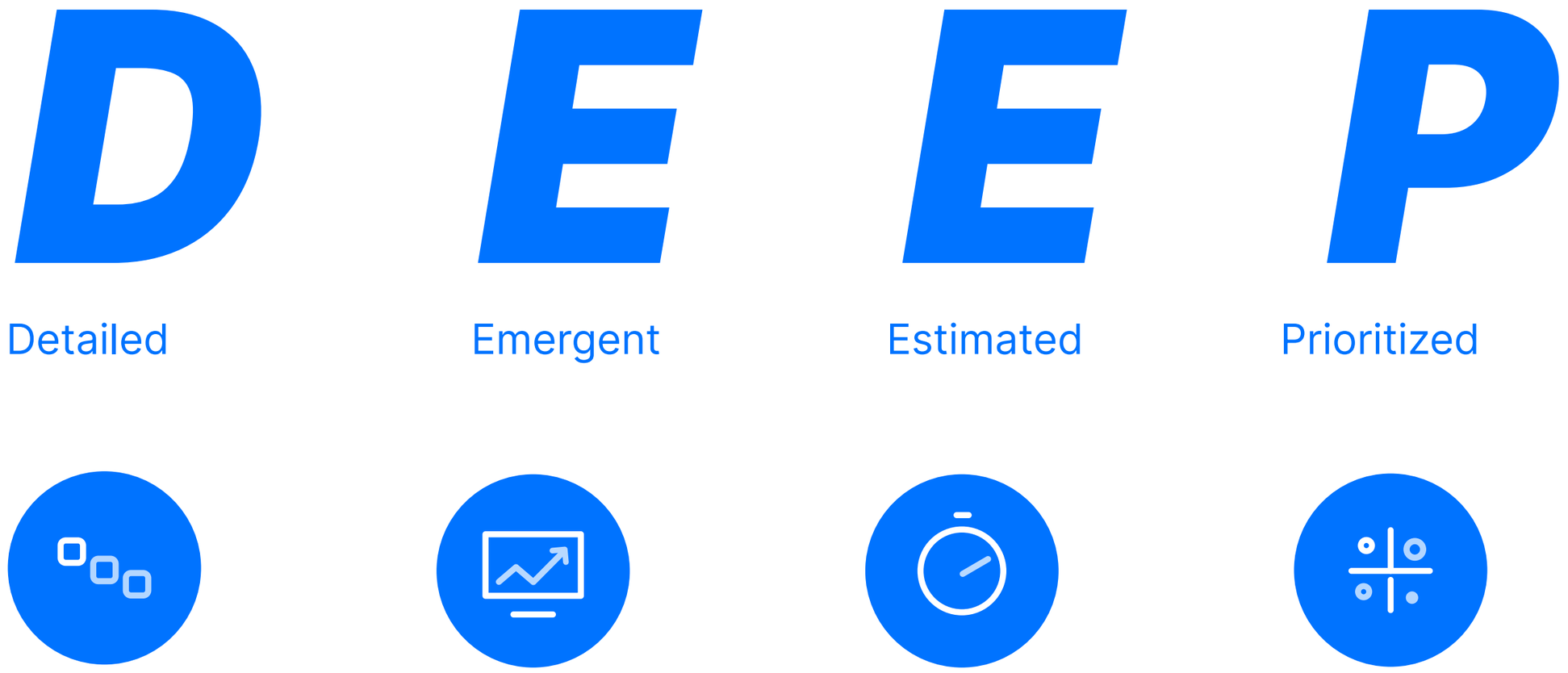
Here are some other things to keep in mind when managing a development backlog:

- Review the backlog on a regular basis and ensure that it's prioritized and the tickets are clear

- Consider grouping items on the backlog into short-term vs. long-term

- Remove items which will never be worked on from the backlog

- Surface related valuable information for each ticket on the backlog for the benefit of your team. For example surfacing the specific customers that are requesting the feature on the main view. This can be done with a development tool like Jira



Now that we have a better understanding of the development backlog let's now discuss **the 5 Scrum rituals in detail.**

1

Story time

This ritual is led by the product owner (the product manager in a Scrum team) and the entire team participates.

The purpose of this meeting is to ensure that all of the tickets that will go into the sprint are ready for development.

In this meeting the product owner will review the tickets, user stories and acceptance criteria with the team, and ensure that the team is comfortable to start working on them.

Story points may also be added to the tickets during this meeting.

A story point is a number that indicates the level of difficulty for fulfilling a piece of work. While it is not meant to denote the duration it takes to complete the work there are many teams that use them for this purpose.

Remember, Scrum can be flexible when needed. It does not have to be, nor should it be, treated as rigid as Waterfall.

One common method that's used for setting story points is the Fibonacci sequence. The scale goes: 1, 2, 3, 5, 8, 13, 21, and so on with each increase in the scale denoting increasing difficulty in the work at hand.

You can read more about this estimation method here.

Some teams decide to assign story points during this meeting while other teams may give their team the needed information for each ticket and then allow them to follow up later with the story points.

This is beneficial especially if they have some research to perform or need to review the codebase to provide more accurate estimates.

How does a team know which number to apply to a ticket?

They determine this based on their past experience. However, keep in mind that it is an estimation.

As time progresses the estimates should become more accurate. And this is one of the benefits of velocity and burndown charts.

2

Sprint planning

The sprint planning meeting is run by the scrum master and the entire Scrum team participates.

Prior to this meeting all of the tickets that are stated to go into the next sprint (meaning they are prioritized at the very top of the backlog) should be fully understood by the team and have story points attached to them.

The main purpose of this meeting is to define the work that will be performed in the sprint. This includes setting the sprint goal. The sprint goal comes from the product owner with input from the team.

Once the sprint goal is defined and the sprint begins the entire Scrum team's focus and energy for that sprint goes towards accomplishing the set goal(s).

Along with setting the goal the team will determine which tickets they will work on and the total workload based on the total story points.

Check out our actionable guide for product managers to run effective sprint planning meetings remotely. The end of this meeting formally begins the sprint.

3

Daily scrum (standup)

Another term for daily scrum is daily standup. These are daily meetings that generally occur in the morning with the entire scrum team.

In these meetings each scrum member provides a summary of their work answering three questions:

- What did you accomplish yesterday?
- What are you focused on today?
- Do you have any blockers?

A blocker is anything that is preventing someone from continuing their work.

These meetings are meant to be quick and informative. Each member of the team provides a quick update giving everyone on the team visibility on what each person is working on and how their work is progressing.

Team members can follow up with those who have blockers after the meeting to provide assistance.

One of the common reasons why these meetings take place standing up (hence the term "standup") is because they are meant to be quick.

People generally don't enjoy standing for long periods of time so if everyone is standing during these meetings then they are more likely to be brief.

I have participated in standups that took longer than 20 minutes because everyone was sitting. Rather than being brief and to the point team members would also get into deep discussions on points mentioned by others

The Scrum master should take note of this and put a stop to it. Detailed discussions should happen after the meeting.

Daily standups are not specific to Scrum. They can take place in Kanban and Waterfall software development as well.

While many teams have these meetings first thing in the morning it does not have to take place at this time. Feel free to modify the time and format in a way that works for your team.

Some teams prefer to have the entire team meet in one central location in the office while others may decide to have everyone hop on a Zoom call. There are even some teams who simply type their standup notes in Slack.

Scrum is not meant to be super prescriptive like Waterfall. Remember the Agile values.

4

Sprint review

During the sprint the team performs the work required to meet the sprint goal.

They share blockers, ask for assistance when needed, and eventually reach the end of their sprint.

The sprint review meeting takes place at the end of the sprint. The main purpose of this meeting is to get feedback from main stakeholders.

The team also gets feedback from the product owner who tests the latest build to ensure that it meets the definition of done (all of the acceptance criteria are met).

If everything is acceptable then the product owner will give their approval and the team will then prepare for release. If there are issues that need to be addressed then they will be pointed out and the team may have some further work to do.

From this meeting the team will have a good understanding of what should be incorporated into the planned release and the product owner will define some items that should be transferred to the following sprint.

This meeting generally ends the sprint. The retrospective meeting which happens after this is a discussion on how this sprint went.

5

Retrospective

The retrospective meeting is when the entire Scrum team gets together to uncover ways to perform better as a team. Every member of the scrum team takes part in this meeting.

This meeting is run by the scrum master. Three main points are discussed regarding the recently completed sprint:

- What went well during this sprint?
- What went poorly during this sprint?
- What action items do we need to take to improve moving forward?

The scrum master takes notes during this meeting, especially of the points mentioned for the last two questions.

When Scrum teams take the retrospective seriously they are able to improve their work and processes with each successive sprint.

Downsides of Scrum

Now that we have a better understanding of Scrum and how it works in detail let's discuss some of its downsides.

Tasking on product managers

Following Scrum can be tasking on product managers, especially when they don't have a dedicated product owner that they work with.

In some companies product managers focus on product strategy while they have dedicated product owners to work with the Scrum team to translate the product strategy into delivery.

Without a scrum master the product manager will be busy addressing both product strategy and delivery work for the product.

Product managers will have various tasks to perform.

Along with refining the roadmap, working with stakeholders to support and enable them, regularly interfacing with customers, and reporting on strategic initiatives, while following Scrum they will also be busy writing user stories and acceptance criteria, performing acceptance testing, and attending the scrum rituals while working with the scrum master to facilitate the scrum process.

Excessive prescription

This is not a downside of Scrum per say but rather on how some teams implement it.

Some Scrum practitioners focus more on the Scrum processes than on the individuals they work with and the goals that they set out to achieve.

As a result change is limited to better planning rather than working to ship valuable work quickly and obtain feedback. Process is beneficial however too much process can be detrimental.

It's important to remember that Scrum is a framework. It's not meant to be as prescriptive as Waterfall and it should not be treated so rigidly.

If and when certain rituals do not work for your team then feel free to modify them (with justification).

We have covered Waterfall, Agile, and Scrum in this guide. Now let's discuss the second most popular way in which Agile software development is implemented.

Kanban

Kanban is a framework for implementing Agile software development. It was developed by Toyota. Yes, the auto manufacturing company.

Toyota developed Kanban after studying how supermarkets stocked shelves.

They noticed that supermarkets had the goal of being “just in time”, meaning that the stocks aren’t continuously overloaded with expired or wasted food. Rather they would be stocked in time with fresh food for consumers to purchase.

Over the years Toyota refined this process for building their cars and now it is used in the software industry to build software.

With Kanban the rules are not as “strict” as Scrum. By this I mean that there aren’t as many defined rituals.

One key concept with Kanban that differentiates it from Scrum is that there are no sprints. Rather, **Kanban uses cycles.**

With Kanban a team will have a defined schedule for when they release new working code (for example every 2nd Monday). Once the date arrives everything that is completed by that date (falls within the “Done” column) gets launched.

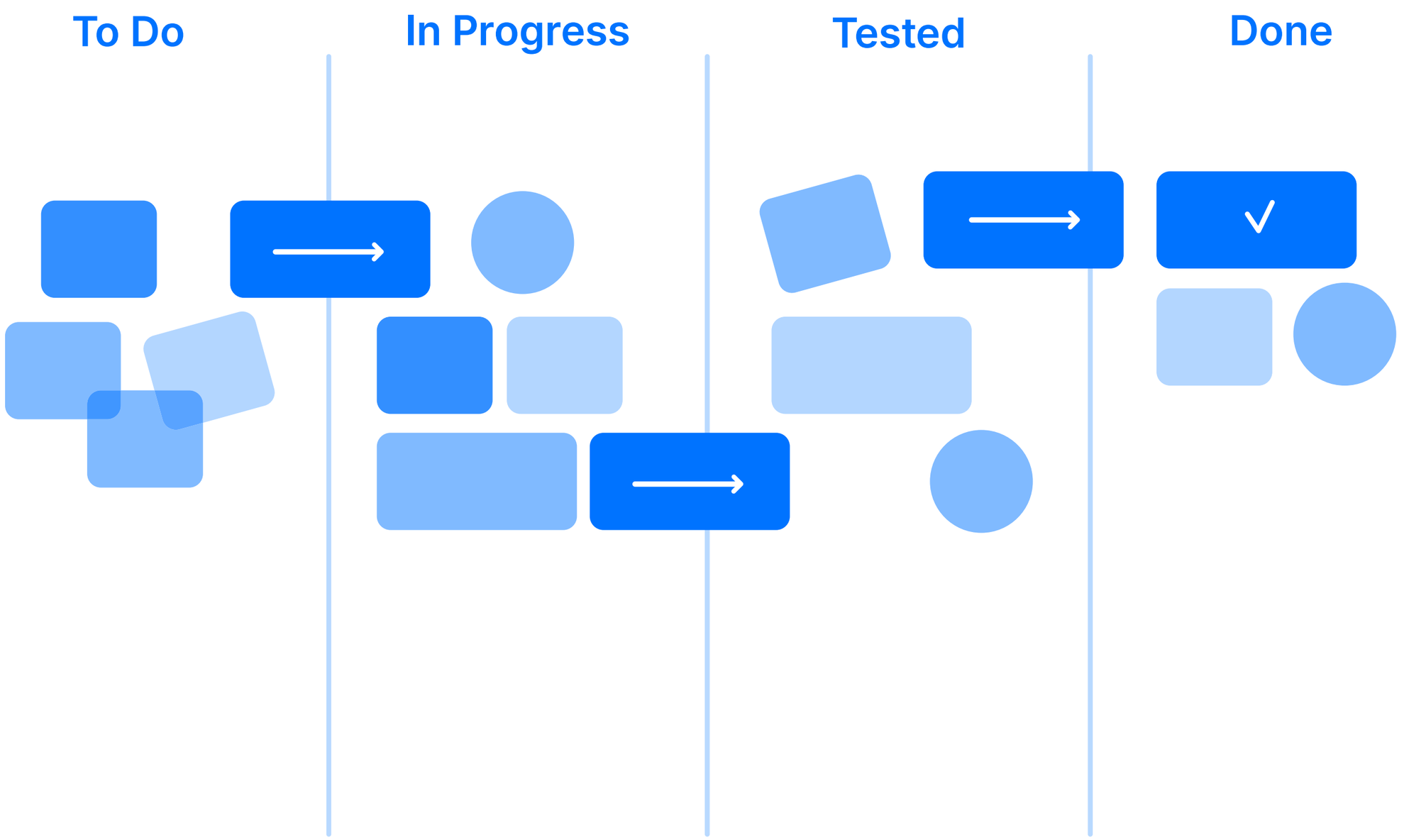
The Kanban board

The main tool that is used in Kanban is the Kanban board. A Kanban board has 4 key columns

1. To do
2. In progress
3. Tested
4. Done

You may see variations of these columns as some companies and teams will name them differently or add additional columns. This is 100% ok, and in fact encouraged if needed. Modify the processes as needed to fit your team's needs.

Within each column are tickets with specifications of the work that needs to be done.



So how else does Kanban work?

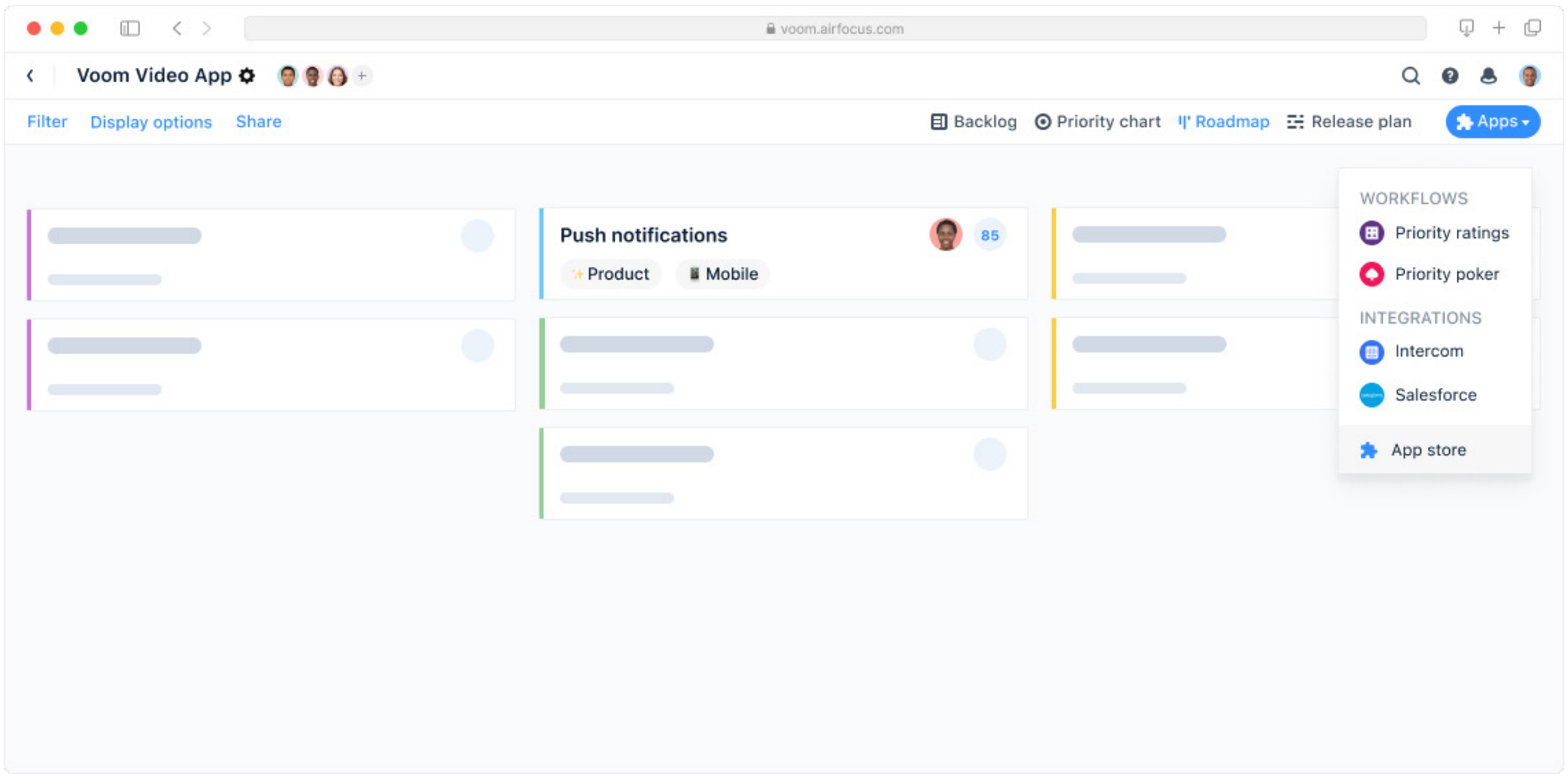
With Kanban only a certain number of tickets can be in progress at a time. It is up to the team to determine the number of tickets. With time and experience a team will know how many they can handle.

When working in Kanban the implementation team will pick up a ticket from the To do list, work on it, and move it along the columns as it progresses.

So a ticket such as “Enable login with Facebook” will begin in the Backlog (or the To Do column) and work it’s way across to Done once it’s complete.

Backlog > In progress > Tested > Done

Once a developer has moved their ticket to done they then pick up their next item from the backlog. And the cycle continues.



Let’s create your Kanban board now - start trial version today and get airfocus 14 days for free

There aren't any particular meetings that are prescribed with kanban as there are for Scrum. However a team can set recurring meetings if they need to, especially if they are implementing Scrumban (Scrum processes mixed with Kanban).

While working with Kanban a product manager will be responsible for working with their development team to prioritize the backlog and ensure that the most important items sit at the top. This way when a developer completes their work they have clarity on what to work on next.

Kanban focuses on the cycle time.

The cycle time is the amount of time it takes a developer to complete their work (from picking it up In Progress to Done).

While the cycle time is measured and teams constantly strive to increase their output, it's important to note that there are many things that can impact the cycle time.

For example if a developer does not have domain knowledge on how to perform a particular feature and needs time to perform research then this will lead to an increase in the cycle time. One way to address this is by creating separate tickets for the research portion of this work.

Benefits and downsides of Kanban

Similar to Scrum, Kanban promotes and **enables continuous development.**

Once the deadline is reached for the next release then all of the work that falls within the Done column gets shipped. If there are items that are close to complete but are not 100% ready then they are not included in the release.

This enables continuous delivery and immediate customer feedback as customers will continue to receive an improved solution at set intervals.

While Kanban is definitely more relaxed than Scrum it does pose some challenges when it comes to evaluating the time required to develop each item. However with the right tools tracking this information is much easier.

Kanban can be beneficial for development teams because it provides more flexibility.

Since Kanban has fewer meetings than Scrum it also gives developers time to focus on solving hard problems.

However Kanban can be tasking on product managers.

Similar to Scrum, product managers need to constantly groom the backlog. Even more so with Kanban because once developers have completed their tasks and are ready to work on the next item they need to know what the next highest priority item is.

The backlog should always be groomed to have the highest priority item at the top. Product managers should not become bottlenecks for their teams.

Another difference between Scrum and Kanban is that in a Scrum team when a developer completes their task ahead of time they may use their additional time to assist others on their team (rather than move on to their next ticket).

This is because the entire team works towards accomplishing the sprint goal(s). So unless there's a good reason to do so they won't pick up another item from the backlog until the next sprint formally begins.

In Kanban however when a developer completes their current ticket they pick up the next priority ticket on the backlog and begin working on it.

When to use Kanban

When should Kanban be used as opposed to Scrum or any other Agile framework?

Kanban is a great option for teams that are less concerned about estimations.

Likewise, for teams that are mainly concerned about getting things done quickly and launching as soon as possible.

For example a development team whose main focus is addressing customer support issues and bugs.

Rather than following the Scrum processes and taking the time to obtain detailed estimates for each ticket and following all of the Scrum rituals, use a Kanban board to track the required work and address the bugs as soon as possible.

Agile Training

Is Agile training beneficial? And for those who are interested in Agile training, where are the best places to get certified?

There are two angles to approach this from.

If you are an aspiring Scrum master or Agile coach then Agile training is definitely required.

For someone who is a new product manager however (recently landed a product role) than achieving Agile training and certification is beneficial.

Taking a reputable course will teach one the ins and outs of various Agile frameworks, processes, and the details on when and how to apply them.

For experienced product managers however, Agile certification is not needed because one will likely gain the knowledge and experience with some of the key Agile frameworks while on the job.

As we mentioned in The Ultimate Guide to a Product Manager's Job, always continue to learn. So read books on this topic, watch videos, attend conferences, and have discussions with others.

And as you learn more continue to apply what you learn and apply it to your work with your teams.

For those who want to obtain Agile certification here are some of the options that are available.

Scrum.org

Scrum.org offers 8 different professional Scrum certifications. These include Professional Scrum Master (PMS), Professional Scrum Product Owner (PSPO), Scaled Professional Scrum (SP), and more.

They have a comprehensive catalog of professional Scrum training that students can enroll in.

To obtain certification students are required to complete one of their courses that they offer throughout the year, or take a course with one of their professional Scrum trainers.

The Scrum Alliance

The Scrum Alliance not only offers certification but also provides a community of Scrum practitioners.

They have a number of certifications (9 in total at the time of authoring this guide) including Certified ScrumMaster, Certified Scrum product Owner, and some more advanced ones that they offer for those interested in Scrum training.

To be awarded a certificate students must successfully pass the final exam at the end of the course.

The Scrum Alliance also requires their students to renew their certificate every two years as well.

SAFe

There is also the SAFe certification that is offered by Scaled Agile.

SAFe is specifically meant for helping larger organizations deal with the challenges that come with practicing Agile at their size and scale.

There are multiple certifications that they offer (13 at the time of this guide) including the Certified SAFe Agile Product Manager, Certified SAFe Scrum Master, Certified SAFe Agilist, and more.

To achieve a certificate students need to enroll and complete one of their courses.

These aren’t the only avenues to earn a certificate. Trainers who are verified by these institutions also provide their own training via other venues throughout the year.

Product management bootcamps

Another avenue to learn agile product management is by enrolling in a product management bootcamp. Product Hall and General Assembly cover Agile frameworks as part of their curriculum.

Though the certificate offered at the end of the course is specifically for product management, graduates will have the knowledge to speak to Agile principles and frameworks, their benefits, pros and cons of various frameworks, and more.

Agile tools

There are various tools that teams use to facilitate Agile processes for software development.

While many of these tools are geared towards project management there are some, like airfocus, that are comprehensive and focus on strategy as well.

Agile tools assist with delivering working software to customers quickly to obtain actionable insights while maintaining product quality.

The list of Agile tools is extensive but here are some of the more popular ones that you can check out.

- airfocus
- Trello
- Jira
- Kanbanize
- Pivotal Tracker
- Clarizen

Check out our article on The Best Agile Project Management Software in 2020 to learn more about some of these tools.

Additional Agile frameworks

Scrum and Kanban are two of the most popular Agile software development frameworks for implementing Agile, however there are others that you should be aware of.

Scaled Agile Framework (SAFe)

SAFe is a popular Agile framework employed by larger organizations.

It is a very flexible framework, mainly due to the fact that it isn't one fixed framework but rather borrows the most successful components from other Agile frameworks.

SAFe is a great framework to utilize for larger organizations that have trouble scaling their Agile teams and dealing with the fast pace of the software industry.

It enables companies to choose the best options of various Agile frameworks that meet their specific needs.

Among its benefits include enabling large companies with many teams to leverage Scrum, Kanban, and other Agile frameworks as they scale.

It also assists with obtaining quick feedback from customers and users and better engagement among stakeholders (clients and the teams that implement the work) which ultimately leads to better quality software and products.

Check out the airfocus glossary to read more about SAFe and its benefits.

eXtreme Programming (XP)

eXtreme programming was introduced in 1996 by American software engineer Kent Beck while he was working on the Chrysler Comprehensive System.

It's an Agile development framework that focuses heavily on producing high quality software that meets customers and user needs, while also improving the development experience for developers.

XP has key practices and rules for engineers that help accomplish these two goals.

For every XP project there are 5 key rules:

1. Planning
2. Management
3. Designing
4. Coding
5. Testing

Each of these rules have set guidelines that developers should follow.

For example small and frequent releases for real feedback from users during the planning phase, cross disciplined team members and rotations to avoid having exclusive specialists for parts of the project, and comprehensive unit tests for the codebase.

Learn more about eXtreme programming here.

Lean Software Development (LSD)

The LSD framework is based on Agile principles.

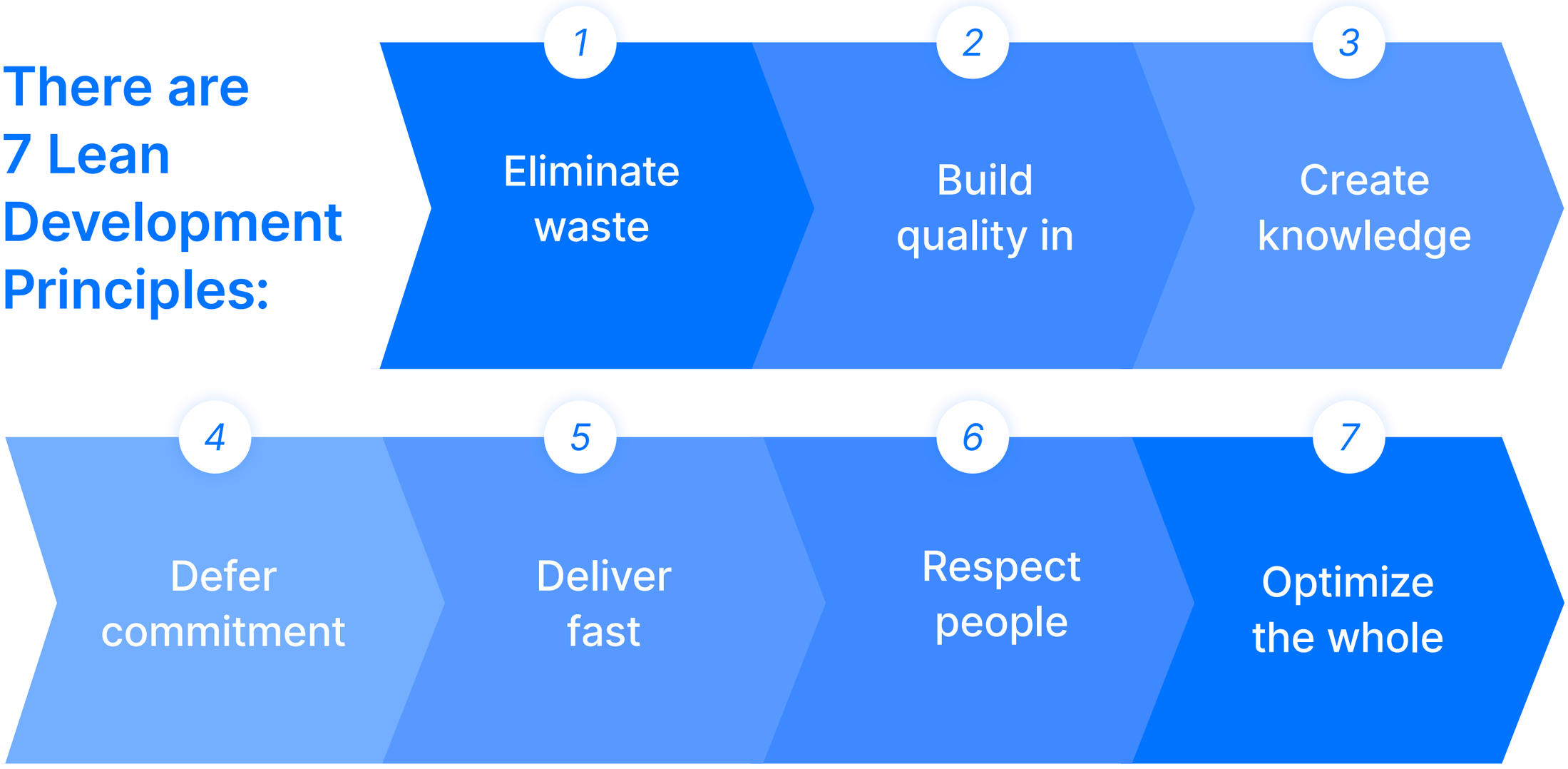
It is used by teams to streamline the development process, the same way that car manufacturers streamline their manufacturing processes (where it originated from).

Where does it come from? Well, Toyota is actually credited with this framework.
In fact when it was first introduced some coined it as the “Toyota Production System”.

LSD was introduced in software development in 2003 in the same year that the book Lean Software Development: An Agile Toolkit was published. This is a highly rated book by Mary Poppendieck that explains how the lean principles from manufacturing offer a better approach to software development. Check it out when you get the chance.

As a framework LSD aims to provide as much value to customers and users as possible and eliminate waste.

There are 7 Lean Development Principles:



Learn more about the principles of LSD here.

Some Agile Best Practices

Here are some of the best practices that are used for Agile software development. These practices are meant to support the Agile frameworks. Following them leads to better code quality and a better product.

Cross-functional team

A cross-functional team is a team that is made up of members that have various skills and perform different functions within a company.

A Scrum team is an example of a cross-functional team as it can include product owners, Scrum masters, product designers, developers, and QA developers.

Team members (possessing various skills) bring their own expertise and ideas to the team which leads to better products (higher quality and innovative solutions).

Continuous integration

Continuous integration (CI) is a practice whereby developers automate the process of integrating the code changes from other developers into one central repository.

CI enables developers to repeatedly merge their changes into their central repository as they improve their product, along with utilizing tests to ensure that there aren't issues with the newly merged code.

Iterative development

To iterate is to repeat something with the goal of achieving a particular goal. With Agile the completion of an iteration begins the following iteration.

Scrum supports iterative development when building solutions. This means that the team defines the solution, releases a working product for customers, and then continues to make changes to the product via iterative development based on customer feedback.

From sprint to sprint (Scrum) or cycle to cycle (Kanban) there is an improvement to the product.

The key benefit of iterative development is that it enables the development team to make incremental improvements to the product, improvements that come from actual feedback from their customers and stakeholders.

Timeboxing

Timeboxing is when a fixed period of time is set for a team, or a team member, to work on completing a specific task or goal.

Instead of allowing the work to continue indefinitely, a fixed period is assigned to the piece of work and the work ends when the time is reached. Along with reviewing what was completed.

There are no hard and fast rules to this, teams can set the time limits that work for their project.

For example a developer may have a time box of 2 days to refactor some code or a team may be given a time box of 2 weeks to focus on resolving as many bugs as they can. Timeboxes can also be limited to minutes as well, for example 45 minutes every morning to accomplish a particular task.

Timeboxing is a key component of Agile and is used heavily with Scrum and Kanban teams.

Pair programming

Pair programming is the act of two developers working together to write code.

While pair programming one developer writes the code (the “driver”) while the other observes the code being written and provides feedback (the “navigator”).

Pair programming is beneficial because it leads to better code quality, helps reduce human error, and can speed up development.

Another benefit of pair programming is that due to the fact that there are two developers that write the code, in the event that one of the developers is unavailable then there is another that has knowledge of the code base and can continue to work on it.

To conclude

Multidisciplinary teams work together to bring the right solutions to life. High quality solutions built with technology that can be placed into the hands of users at the earliest to obtain quick feedback to act upon and provide them with further value.

This ultimate guide covered some of the key frameworks that are used by companies to accomplish this.

Now that you have made it to the end of this detailed guide you should have a much more solid understanding of Agile and its principles. Terms such as Scrum, Kanban, Waterscrum, lean, and more should not sound foreign to you.

And you should be able to clearly articulate the benefits of Agile and how it differs from Waterfall.

One lasting piece of advice before you go.

Remember that these frameworks and practices are guides and they can be modified when needed. When working with Agile teams keep its 12 agile principles in mind.

To learn more about Agile software development and dive deeper into some of the topics mentioned in this guide check out the airfocus blog. It's complete with knowledge, tools, and advice from our team based on the framework used by experienced professionals in the industry.

About airfocus

airfocus offers a modern and modular product management platform. It provides a complete solution for product teams to manage and communicate their strategy, prioritize their work, build roadmaps, and connect feedback to solve the right problems. Designed with flexibility in mind, airfocus allows you to quickly customize the platform to fit your needs without disrupting the way your team works.

Join thousands of global product teams who use airfocus to make better decisions and build outstanding products.

Learn more at airfocus.com and start your 14-day trial now.

airfocus