- Module 1: Introduction to Agile and DevOps
- Module 2: Agile Methodologies

Course Objectives

- By the end of this course, participants will:
- Understand the principles and practices of Agile and DevOps.
- Learn popular Agile methodologies like Scrum and Kanban.
- Master DevOps tools for CI/CD, containerization, infrastructure as code (IaC), and monitoring.
- Be able to implement Agile and DevOps practices in real projects.
- Understand how Agile and DevOps combine for faster, reliable software delivery.

- What is Agile?
- What is the Agile Methodology?
- What is the Agile Manifesto Software Development?
- What are the 12 Agile Principles?
- Life cycle of Agile Methodology
- Benefits of Agile development methodology
- Types of Agile Methodologies
- Agile Software Development
- Agile Development Models
- Agile Software Development Methodology
- What is Agile Product Management?
- What is Agile Project Management?
- Agile Software Testing
- Agile Methodology Advantage and Disadvantage
- Agile vs Waterfall Methodology

What is Agile?

- Agile is an **iterative and incremental approach** to software development.
- Focuses on **collaboration, customer feedback**, and **rapid delivery** of small, functional pieces.
- Originated as a response to rigid, plan-driven traditional approaches.
- Instead of rigid, long-term plans, Agile teams break projects into smaller, manageable tasks **(sprints)** and adapt to changes quickly. This approach allows for continuous feedback, frequent deliveries, and a focus on delivering value to the customer.

What is Agile?

- Agile is a Project Management and software development approach that aims to be more effective.
- It focuses on delivering smaller pieces of work regularly instead of one big launch.
- This allows teams to adapt to changes quickly and provide customer value faster.
- Agile is not just a methodology; it’s a mindset. Agile isn't about following specific rituals or techniques. Instead, it's a bunch of methods that show a dedication to quick feedback and always getting better.
- They prioritize flexibility, collaboration, and customer satisfaction.
- Major companies like Facebook, Google, and Amazon use Agile because of its adaptability and customer-focused approach.

History of Agile

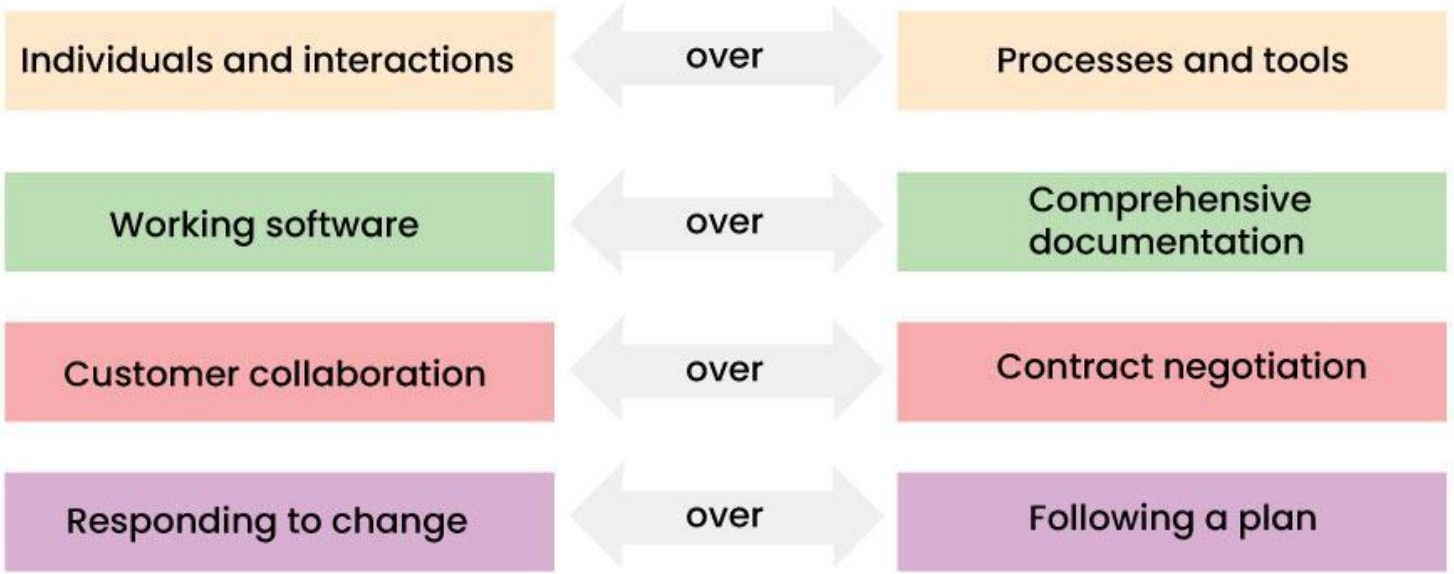
- In 1957, people started figuring out new ways to build computer programs. They wanted to make the process better over time, so they came up with iterative and incremental methods.
- In the 1970s, people started using adaptive software development and evolutionary project management. This means they were adjusting and evolving how they built software.
- In 1990s, there was a big change. Some people didn't like the strict and super-planned ways of doing things in software development. They called these old ways "waterfall." So, in response, lighter and more flexible methods showed up.

What is the Agile Manifesto Software Development?

The **Manifesto for Agile Software Development** is a document produced by 17 developers at Snowbird, Utah in 2001. This document consists of 4 Agile Values and 12 Agile Principles. These 12 principals and 4 agile values provide a guide to Software Developers. The Manifesto for Agile Software Development emerged as a transformative guide to **Software Development**.

The Agile Manifesto 4 Core Values:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan



12 Principles of Agile Manifesto

- **Customer satisfaction** through early and continuous delivery
- **Welcome change**, even late in development
- Deliver **working software frequently**
- Close **collaboration** between business & development
- Build around **motivated individuals**
- Prefer **face-to-face communication**
- Working software is the **primary measure** of progress
- Promote **sustainable development**
- Focus on **technical excellence** and good design
- Simplicity is essential
- **Self-organizing teams** deliver the best results
- Reflect and adjust at regular intervals

Customer Satisfactions

Changing Requirements

Frequent Delivery

Communicate Regularly

Face-to-face communication

Measure work progress

Development process

Good Design

Measure progress

Support team member

Continue Seeking Result

Reflect and adjust regularly

Agile Mindset in Action

- Value over process: Emphasize **outcomes** over formal rules
- Adaptability: **Respond to change** quickly and effectively
- Collaboration: Encourage **open communication and trust**
- Iterative: Deliver in **small, usable increments**

- 📌 Think: **Learn, Adapt, Deliver, Repeat**

Why is the Agile Manifesto important?

- It is a revolutionized Software Development that focuses on a more flexible and Adaptive approach rather than a rigid approach.

- It enhances Team collaboration where individuals can interact openly and freely, also agile focuses on a collaborative work environment where team members can share their ideas openly, communicate with each other and work efficiently.

- It helps in increasing customer satisfaction, Customers are also involved in the process which ensures the product aligns with their needs and expectations, leading to higher satisfaction and a better end product.

- Promote Flexibility and Adaptability, agile allows for 'responding to change', means allows teams to adapt to evolving requirements and market dynamics and reduces the risk of project failure.

- Accelerate Delivery, by focusing on working software, Agile encourages iterative development and frequent releases, enabling faster delivery of valuable features to customers.

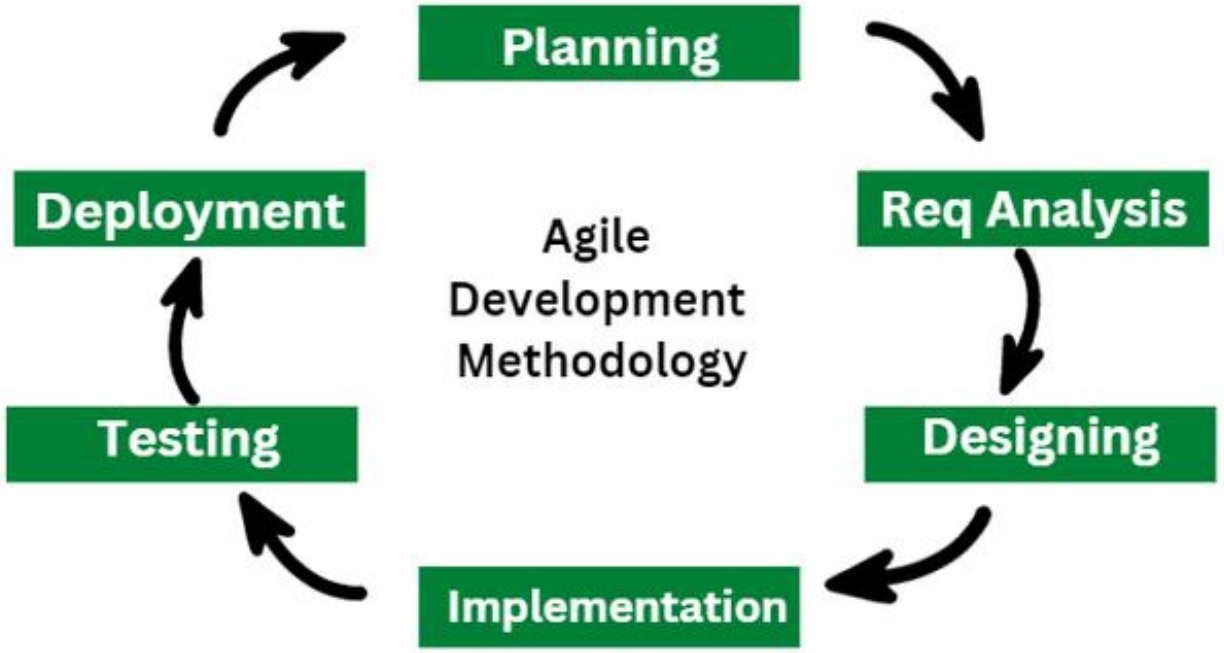
How to use the Agile Manifesto?

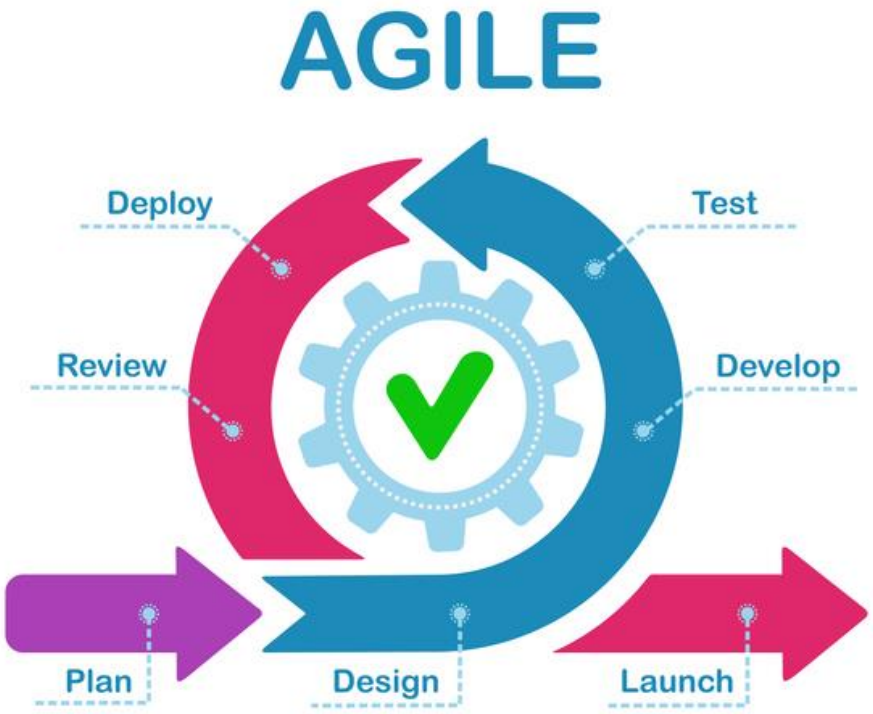
How an organization shifts its culture from a rigid one to a more flexible approach.

Here are some steps to use the Agile Manifesto

- **Focuses on Agile Core Values** first by encouraging your team, they can internalize and prioritize the four core values in their daily work.
- **Adopt Agile Frameworks** Implement Agile frameworks like Scrum, Kanban, and Extreme Programming that align with your team's needs and project requirements.
- **Foster Open Communication, and**, encourage regular communication among team members and stakeholders freely, allowing them to share their ideas by creating an environment where team members feel comfortable.
- **Iterate and Improve,** according to change in response, continuously assess your process, and as per the feedback from the user make changes and adjustments. Agile is about continuous improvement and learning.
- **Engage with Customers** Maintain ongoing collaboration with customers to gather feedback and ensure the product aligns with their needs.

Life cycle of Agile Methodology





Life cycle of Agile Methodology

1. Requirement Gathering

- In this stage, the project team identifies and documents the needs and expectations of various stakeholders, including clients, users, and subject matter experts.
- It involves defining the **Project's Scope**, objectives, and requirements.
- Establishing a budget and schedule.
- Creating a project plan and allocating resources.

2. Design

- Developing a high-level system architecture.
- Creating detailed specifications, which include data structures, algorithms, and interfaces.
- Planning for the software's user interface.

- **3. Development (Coding)**

Life cycle of Agile Methodology

3. Development (Coding)

- Writing the actual code for the software.
- Conducting unit testing to verify the functionality of individual components.

4. Testing

- This phase involves several types of testing:
- **Integration Testing:** Ensuring that different components work together.
- **System Testing:** Testing the entire system as a whole.
- **User Acceptance Testing**: Confirming that the software meets user requirements.
- **Performance Testing:** Assessing the system's speed, scalability, and stability.

Life cycle of Agile Methodology

5. Deployment

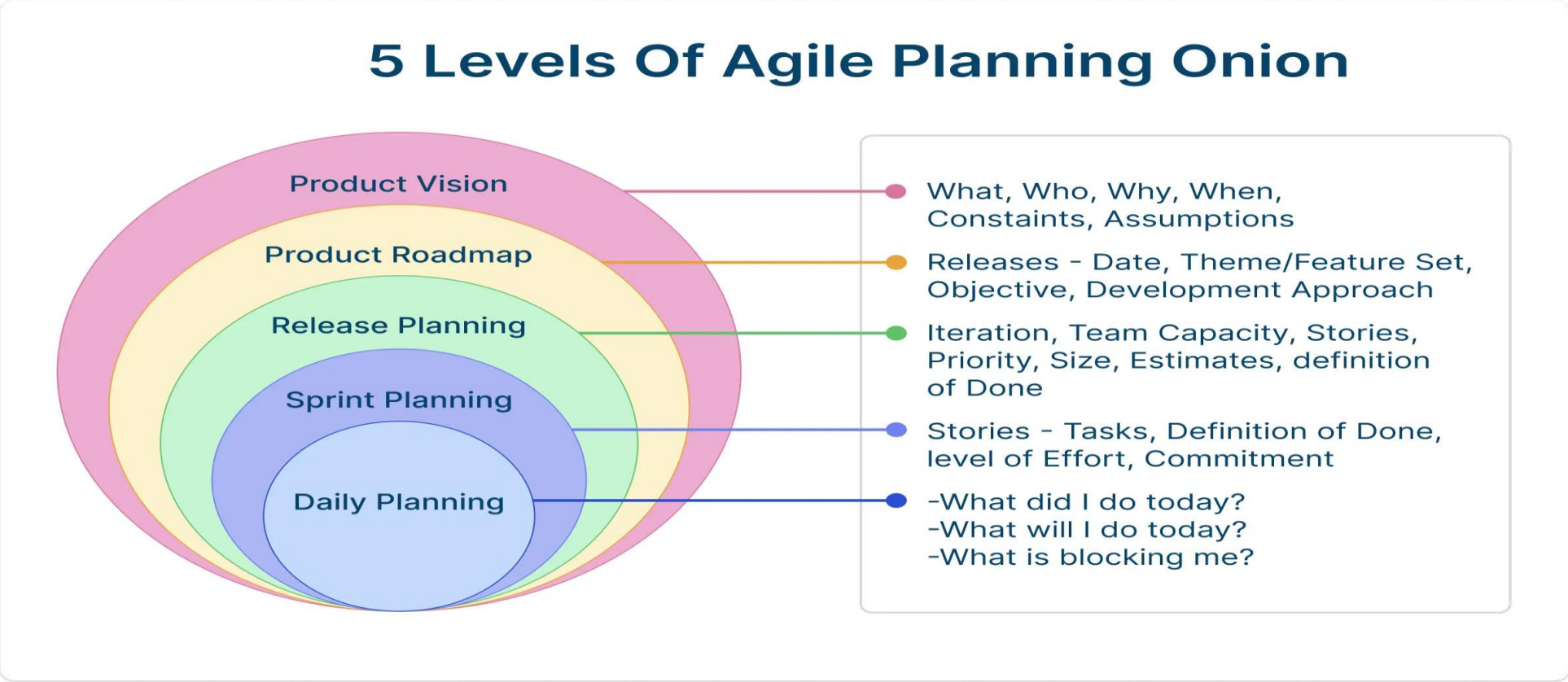
- Deploying the software to a production environment.
- Put the software into the real world where people can use it.
- Make sure it works smoothly in the real world.
- Providing training and support for end-users.

6. Review (Maintenance)

- Addressing and resolving any issues that may arise after deployment.
- Releasing updates and patches to enhance the software and address problems.

Agile Planning Onion

- The Agile Planning Onion represents different levels of planning in Agile, balancing long-term strategy with short-term execution.



Agile Planning Onion

It consists of five interconnected layers, each contributing to a structured yet flexible planning approach:

- **Vision** – Defines the long-term business objectives and market impact of the product.
- **Roadmap** – Outlines the high-level strategy, key milestones, and directional goals over time.
- **Release Planning** – Plans for delivering functional increments of the product in scheduled releases.
- **Iteration Planning** – Focuses on short-term sprints, selecting tasks from the backlog.
- **Daily Planning** – Aligns team members through stand-ups, tracking progress, and addressing blockers.

Advantages of Agile

- **Flexibility and Adaptability**: Agile can quickly adapt to changes, allowing teams to respond to new customer needs and market conditions.
- **Improved Collaboration**: Agile encourages constant communication between developers and stakeholders, ensuring the product meets user expectations.
- **Faster Delivery**: Agile ensures quicker releases, keeping customers engaged and their feedback incorporated early.
- **Enhanced Quality and Customer Satisfaction**: Agile focuses on customer feedback, ensuring the product meets their needs and delivering high-quality results.
- **Iterative Development**: Work is done in small, manageable steps, allowing for regular improvements and quick adjustments.
- **Transparency**: Agile keeps stakeholders informed at every stage, ensuring clarity and alignment.
- **Quality Assurance**: Agile prioritizes quality, ensuring the product meets users' expectations through continuous improvements.
- **Continuous Improvement**: Regular feedback ensures the product keeps improving, preventing last-minute issues and maintaining high quality.

Types of Agile Methodologies

1. Kanban

- Kanban Method is an approach to evolutionary and incremental systems and process change for organizations. A work-in-progress limited pull system is the central mechanism to uncover system operation (or process) complications and encourage collaboration to continuously improve the system.

Kanban is best suited in the below scenarios:

- Dynamic/ frequent changing requirements which need to be delivered faster.
- In case of changing priorities, the team can pull the prioritized work as soon as the WIP limit drops.
- Frequent releases are there (Periodically).
- When incoming work is continuous.

2. Scrum

- Scrum is great for small teams and works in **sprints**—short, focused work periods. A **Scrum Master** removes obstacles for the team. Scrum includes two key events:
- **Sprint Planning:** Decides what the team will work on in the next sprint.
- **Sprint Retrospective:** Reflects on the last sprint to improve the process for the next one.

3. Extreme Programming (XP)

Extreme Programming (XP) focuses on technical practices and rapid delivery. It emphasizes values like **communication, simplicity, feedback, courage**, and **respect**. XP helps teams release updates frequently, making it ideal for projects where customer feedback and fast changes are crucial.

4. Adaptive Project Framework (APF)

Adaptive Project Framework (APF) is designed for projects where things can change unexpectedly. It's often used in IT projects where traditional methods may not apply. APF focuses on adapting to changes in resources, budgets, timelines, or team members as the project progresses.

5. Extreme Project Management (XPM)

Extreme Project Management (XPM) is used for complex projects with a lot of uncertainty. It’s highly flexible, allowing teams to constantly adjust their processes and strategies. XPM involves short **sprints** for rapid problem-solving and adjusting approaches as needed.

6. Adaptive Software Development (ASD)

- Adaptive Software Development (ASD) is about continuous learning and adaptation. The three main phases are:
- **Speculate:** Develop ideas about how the project should progress.
- **Collaborate:** Work closely with the team and customers.
- **Learn:** Use feedback to improve and make better decisions.
  This approach is fluid, allowing teams to work through all phases simultaneously and adjust quickly.

7. Dynamic Systems Development Method (DSDM)

- Dynamic Systems Development Method (DSDM) is more structured and covers the entire project lifecycle. It includes four main phases:
- **Feasibility and Business Study:** Understand the project's goals and its feasibility.
- **Functional Mode or Prototype Iteration:** Build prototypes to refine the solution.
- **Design and Build Iteration:** Develop the final solution.
- **Implementation:** Deliver the completed product.
  DSDM is ideal for projects needing a solid structure while still maintaining flexibility.

8. Feature Driven Development (FDD)

Feature Driven Development (FDD) focuses on developing specific features. The team works on one feature at a time, delivering regular updates based on customer feedback. It allows for fast fixes and frequent progress, keeping the project moving forward.

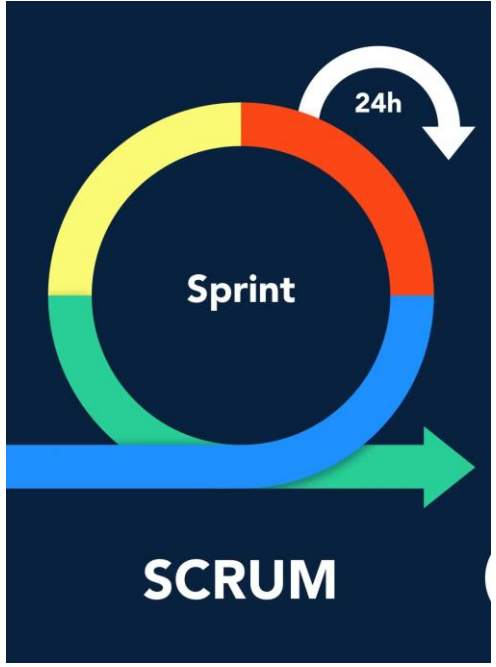
Parameters	Agile Methodology	Traditional Approach
Definition	Agile is like building a flexible and adaptable treehouse in stages.	Traditional approaches are like constructing a house with a detailed blueprint.
Chronology of operations	Testing and development processes are performed concurrently.	Testing is done once the development phase is completed.
Organizational structure	It follows iterative organizational structure.	It follows linear organizational structure.
Communication	Agile encourages face-to-face communication.	Traditional approach encourages formal communication.
Number of phases	It consists of only three phases.	It consists of five phases.
Development cost	Less using this methodology.	More using this methodology.
User requirements	Clearly defined user requirements before coding.	Requires interactive user inputs.

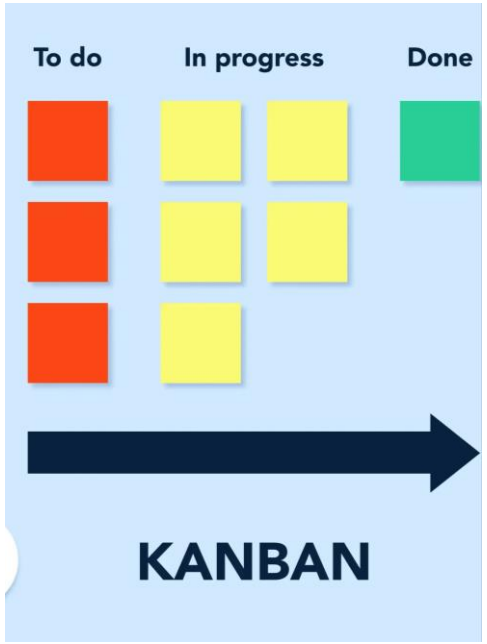
When to go for Agile?

- **Unclear or Changing Requirements:** Agile is great for projects with requirements that aren't well-defined or might change.
- **Complex Projects:** It's good for big, complex projects by breaking them into smaller pieces.
- **Customer Focus:** Use Agile when making customers happy is a priority and you want to involve them throughout.
- **Quick Time-to-Market:** If you need to get your product out fast, Agile can help.
- **Small to Medium Teams:** Agile works well for teams of a few to a few dozen people.
- **Team Skills:** It's best when you have a mix of skills in your team, like development, testing, design, and more.
- **Collaboration:** Agile promotes working together and open communication.
- **Regular Updates:** If you want to check progress often and make changes as needed.
- **Transparency:** Agile emphasizes being open and clear with everyone involved in the project.
- **Risk Control:** It helps manage risks by tackling issues as they come up.
- **Innovation:** If you encourage trying new things and learning from experience, Agile supports that.
- **Continuous Improvement:** Agile fosters a culture of always getting better over time.

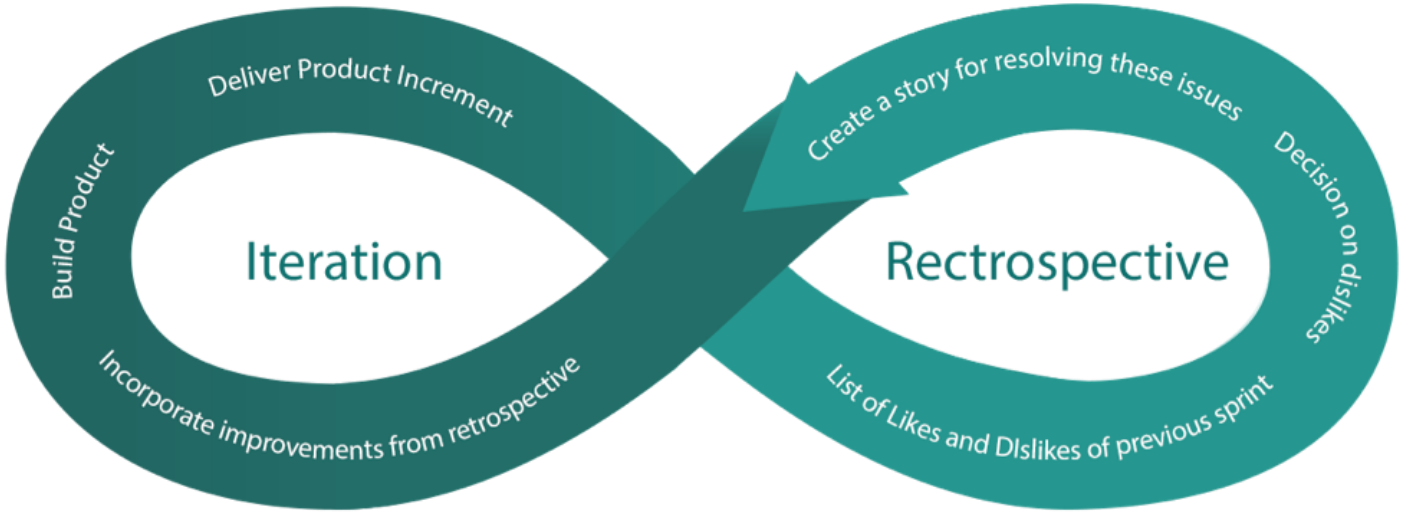
Scrum Vs Kanban

Methodology	Kanban	Scrum
Roles	No defined roles	Scrum master, product owner, and development team
Delivery cycle	Continuous	Sprint cycle lasts one to four weeks
Change policy	Can be incorporated any time	Generally not made during sprint
Artifacts	Kanban board	Product backlog, sprint backlog, product increments
Tools	Jira Software, Kanbanize, SwiftKanban, Trello, Asana	Jira Software, Axosoft, VivifyScrum, Targetprocess
Key concepts or pillars	Effective, efficient, predictable	Transparency, adaptation, inspection



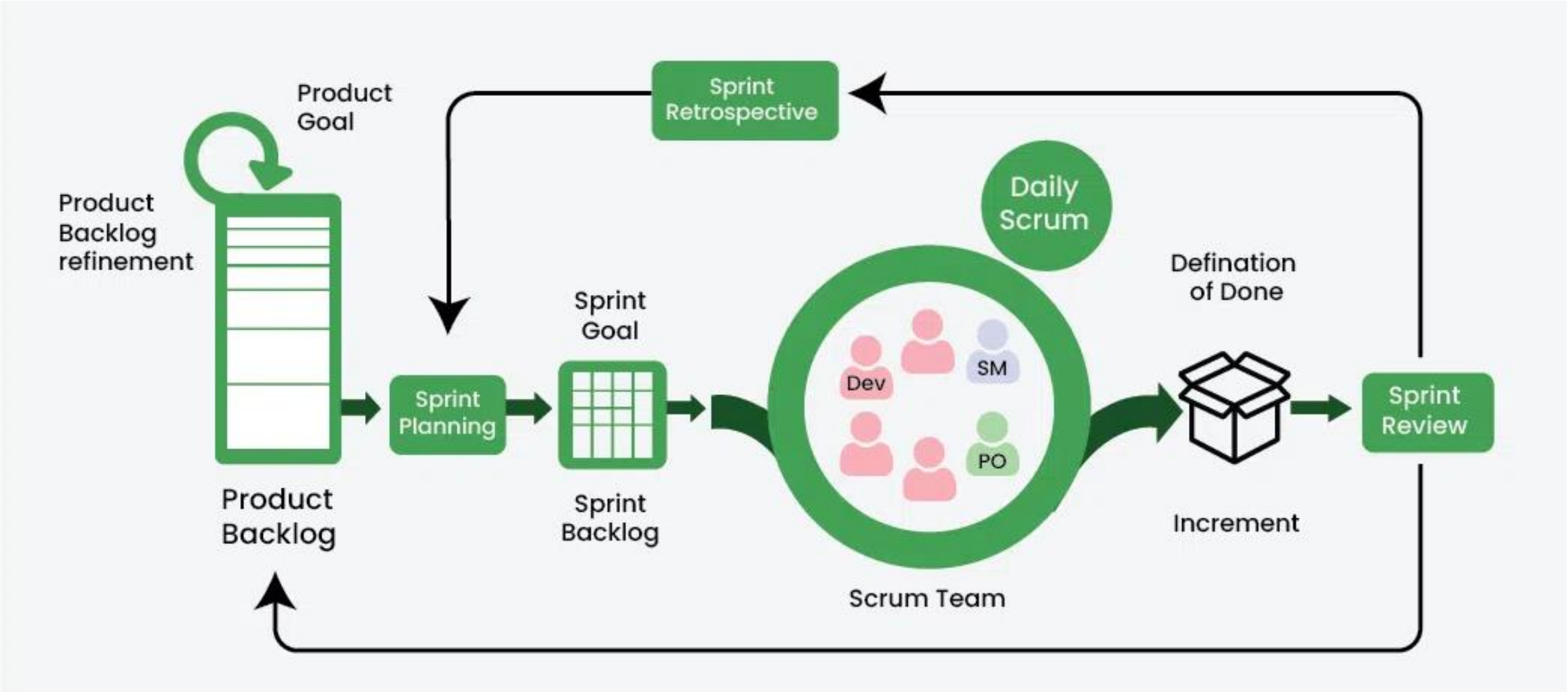


Agile Retrospective



- The Agile retrospective is a unique opportunity for the team to evaluate and improve its *process*. It is a time-boxed meeting headed by the Scrum Master which aims to evaluate a recently ended sprint. Its aim is to find out the factors which can make the next sprint more productive and/or more enjoyable.

Scrum Framework



What is Scrum?

- Instead of doing everything at once, Scrum breaks work into smaller parts called "sprints.“
- Each sprint focuses on completing a specific piece of the project, allowing teams to adapt and improve as they go.
- It's all about teamwork, communication, and getting things done step by step.

What is Scrum?

- Scrum allows us to develop products of the highest value while making sure that we maintain creativity and productivity.
- The iterative and incremental approach used in scrum allows the teams to adapt to the changing requirements.

Scrum Framework:

The Scrum framework is a simple, agile way to manage projects that helps teams work together to create high-quality products quickly.

Scrum Framework Components

Scrum Team

- **Product Owner:** Decides what the team should work on and makes sure it meets business and customer needs.
- **Scrum Master:** Helps the team follow Scrum rules and removes any obstacles they face.
- **Development Team:** A group of people with different skills who work together to build the product.

Scrum Framework Components

Artifacts:

- **Product Backlog**: A list of everything that needs to be done for the product, organized by priority.
- **Sprint Backlog:** A list of tasks the team plans to complete during the current sprint.
- **Increment:** The finished work at the end of a sprint that could be released to users.

Scrum Framework Components

Events:

- **Sprint:** A short, fixed period (usually 1-4 weeks) where the team works on specific tasks from the sprint backlog.
- **Sprint Planning:** A meeting at the start of the sprint where the team decides what to work on.
- **Daily Scrum:** A quick daily meeting where the team checks in on progress and plans the day's work.
- **Sprint Review:** A meeting at the end of the sprint where the team shows what they've accomplished and gets feedback.
- **Sprint Retrospective:** A meeting at the end of the sprint where the team discusses what went well and what could be improved for the next sprint.

Key Terminologies of Scrum

Here are some of key terminologies of Scrum:

- **Product Backlog**: The product backlog is known to be the prioritized list of fixes as well as features that is included in the product's roadmap.
- **Sprint**: Sprint is known as a **time-box** event which typically lasts from one week to four weeks, in this phase a product increment or iteration occurs.
- **Development Team**: The development team is a group of individuals who are professional in their field and are responsible for product delivery.
- **Daily Scrum**: Daily scrum is a 15 minute daily meeting used by the development team to integrate activities and to create a plan for the next 24 hours of development.
- **Sprint Review**: The sprint review is held at the end of the sprint in which the team presents all the work that is completed to their stakeholders and the stakeholders give back their feedback.
- **Sprint Retrospective**: The sprint retrospective is a meeting concluded at the end of each sprint so that the team can discuss what went well and what could be improved as well as how to make those improvements.

Key Terminologies of Scrum

Here are some of key terminologies of Scrum:

- **Product Backlog**: The product backlog is known to be the prioritized list of fixes as well as features that is included in the product's roadmap.
- **Sprint**: Sprint is known as a **time-box** event which typically lasts from one week to four weeks, in this phase a product increment or iteration occurs.
- **Development Team**: The development team is a group of individuals who are professional in their field and are responsible for product delivery.
- **Daily Scrum**: Daily scrum is a 15 minute daily meeting used by the development team to integrate activities and to create a plan for the next 24 hours of development.
- **Sprint Review**: The sprint review is held at the end of the sprint in which the team presents all the work that is completed to their stakeholders and the stakeholders give back their feedback.
- **Sprint Retrospective**: The sprint retrospective is a meeting concluded at the end of each sprint so that the team can discuss what went well and what could be improved as well as how to make those improvements.

Scrum Principles for Project Success

- **Transparency:** When people operate in a team setting, they are aware of the difficulties that other people may be facing. Frequent in-person meetings between project owners and members of cross-functional teams help to avoid misunderstandings and information.
- **Reflection:** The framework includes regular reflection points so that team members can assess their development. Project managers estimate and prepare for the future using the information take from these review meetings. Projects can therefore operate more effectively, on time, and within budget.
- **Adaptation:** Tasks can be rearranged in order by team members in response to evolving client needs. They choose which assignments to finish right away and which to come back to later.

Scrum Values for Project Teams

- **Commitment:** Members of the Scrum Team are devoted to finding the best solution through continuous improvement and time-based tasks and goals.

- **Courage:** By posing direct, difficult questions, Scrum Teams demonstrate bravery. To find the best answer, they have frank and open discussions.

- **Respect:** The Scrum process, the project managers, and each other are respected by the team members. Within the team, this respectful culture fosters a sense of cooperation and mutual support.

Agile at scale
Movin' on up: scaling agile in large organizations

- Real challenge is extending Agile platform across multiple teams in a large organization. In other words, implementing agile at scale.

- Why are companies scaling agile?

Why Scaled?

1. **To Manage Complexity in Large Teams**
   - Agile works well for small teams, but enterprises often have hundreds or thousands of employees working across multiple departments or geographies.
   - Scaling Agile helps coordinate multiple Agile teams working on the same product or program without chaos.

2. **To Improve Time-to-Market**
   - Traditional development models slow down decision-making and delivery.
   - Scaled Agile enables **faster feedback loops**, better prioritization, and more **frequent releases**, helping companies respond quickly to market changes.

Why Scaled?

3. To Align Business and IT

- Agile at scale ensures that **business goals, product development, and IT infrastructure** are aligned.
- Frameworks like SAFe (Scaled Agile Framework) promote **strategic alignment** from executive level to individual teams.

4. To Enhance Collaboration Across Teams

- Cross-functional teams can become siloed as organizations grow.
- Scaled Agile encourages **synchronized planning**, shared backlogs, and continuous integration to keep everyone on the same page.

5. To Deliver Value Continuously

- Agile emphasizes **delivering customer value early and often**.
- At scale, this ensures that **every team contributes directly** to business outcomes, not just isolated tasks.

Why Scaled?

6. To Foster Innovation and Adaptability

- In a competitive environment, companies need to **experiment and adapt quickly**.
- Scaled Agile supports **lean startup principles**, encouraging MVPs, iterative learning, and continuous improvement.

7. To Standardize Practices Across the Organization

- Without scaling, teams may adopt **inconsistent Agile practices**, leading to misalignment.
- Scaled Agile frameworks provide a **common language, cadence, and process** across all teams.

8. To Improve Transparency and Governance

- Enterprise-level projects require tracking, compliance, and reporting.
- Scaled Agile frameworks offer **structured planning**, **program-level visibility**, and **metrics** for better decision-making.

Why Scaled?

9. To Drive Cultural and Digital Transformation

- Agile isn't just a methodology; it’s a **cultural shift**.
- Scaling Agile supports broader **organizational agility**, preparing the enterprise for **digital transformation**, innovation, and resilience.

Examples of Scaled Agile Frameworks:

- SAFe (Scaled Agile Framework)
- LeSS (Large Scale Scrum)
- Disciplined Agile Delivery (DAD)
- Spotify Model

What is “Scaling”?

- When we speak of “scaling”, we refer to any situation where more than one team is asked to work together to build a product or service to deliver value to the customer faster.

- With several teams involved, we face new challenges like the need for increased synchronization and improved communication.

Why are companies scaling agile?

- Today, businesses need to be able to adapt, at enterprise scale, in order to stay competitive.
- The means to do so: responding to customers' evolving needs and delighting them in the process, providing flexible/customizable solutions, supporting teams of teams working on a unified front, shifting mindsets to place technology as a strategic enabler, and inspiring agile ways of working outside of software and IT teams.
- But without a clear-cut plan or framework, it’s increasingly harder for companies that are scaling to predict delivery, manage cross-team dependencies, and focus on the right business objectives.
- As a result, this often leads to a decline in customer satisfaction, loss of market share or revenue, and more.

So, what is agile at scale?

- Agile at scale is the ability to drive agile at the team level, while applying the same sustainable principles, practices, and outcomes at other layers of the organization.
- Scaling agile frameworks is a cultural transformation, where the business' people, practices, and tools are committed to improving collaboration and the organization's ability to execute against its strategy.
- Ultimately, changes across these areas will help decentralize decision-making, create greater transparency and alignment around work, and increase speed to market, all while hard coding the values of agile into the DNA of the organization.

Where are you on your agile at scale journey?

- We like to chart how far along an organization is in its journey of scaling agile frameworks by looking at how teams and individuals are adopting agile practices.
- Organizations at the beginning of their journey may only have pockets of people practicing agile, and work may be dominated by traditional project management procedures focused on managing a project from conception through to delivery.
- No matter where you find yourself today, acknowledge and respect your position, and start from there.

Popular frameworks for scaling agile

- There's no right way to scale agile. But many organizations have had great success evolving their processes, teams, and cultures using frameworks for scaling agile.

Here's a brief overview of the top agile scaling frameworks frameworks to explore:

❑SAFe

- The Scaled Agile Framework® (SAFe®) is a set of organization and workflow patterns for implementing agile practices at enterprise scale. It was formed around three primary bodies of knowledge: agile software development, lean product development, and systems thinking. SAFe promotes alignment, collaboration, and delivery across large numbers of agile teams.

❑LeSS

- Large-Scale Scrum (LeSS) is essentially regular scrum applied to large-scale development. LeSS is based on the idea that scaling frameworks should be minimalistic (i.e. include less rules, roles, and artifacts) to drive success. However, both LeSS and SAFe share some common patterns: Scrum at the team level, many teams sharing a backlog, collaborative planning across multiple teams, along with the general principles of pull and self-organization that any smaller agile team may be familiar with.

Popular frameworks for scaling agile

❑DA

- Disciplined Agile (DA), previously referred to as Disciplined Agile Delivery (DAD), is a learning-oriented process decision framework for IT solution delivery. It provides a solid foundation from which to scale agile solution delivery within enterprise-class organizations. DA utilizes scrum and kanban, along with transformation knowledge in areas like HR and finance, governance, DevOps, portfolio management. and more. DA is often considered more flexible and easier to scale than other methods.

❑Spotify

- Spotify's approach wasn't meant to be a framework per se, but the organization's take on agile has organically emerged as one. The Spotify model is a people-driven, autonomous framework for scaling agile. It stresses the importance of culture and networks and provides an example for dealing with multiple teams in a product development organization.

❑Scrum@Scale (S@S)

- Scrum@Scale is an extension of the scrum framework. Scrum@Scale is generally adopted by organizations that have already implemented scrum successfully at the team level and are looking to spread it throughout the organization. The main goal is to align growing organizations around one common and shared set of goals. Coordination is managed through a Scrum of Scrums, which is comprised of Scrum Masters from each team, and a MetaScrum made up of product owners.

- What is a product roadmap?
- A product roadmap is a shared source of truth that outlines the vision, direction, priorities, and progress of a product over time. It’s a plan of action that aligns the organization around short and long-term goals for the product or project, and how they will be achieved.

7 Key Benefits of a Cross-Functional Agile Team and RPA Center of Excellence (CoE)

1. Cross-Functional Teams Resolve Issues of 'Conflicting Priorities' for RPA and Product Teams
2. Cross-Functional Teams Improve Communication and Quality
3. Cross-Functional Teams Ensure Consistent Focus on the Customer Experience
4. Cross-Functional Teams Iterate Quickly
5. Cross-Functional Teams Improve Conflict Resolution
6. Cross-Functional Teams Improve Alignment and Use of Resources
7. Cross-Functional Teams Lead to Greater RPA and Product Innovation

Some important Links

- https://www.atlassian.com/agile/manifesto
- Agile: https://www.youtube.com/watch?v=oTZd2vo3FQU
- ROVO: https://www.youtube.com/watch?v=I1R7MY1HQzE
- Agile Scrum Master Complete:
  https://www.youtube.com/watch?v=XwbEZeo31wA

Go through Jira and Scrum documentation