AGILE METHODOLOGY

Mansi Thakur

Table of Contents

1. What is Agile Methodology?

Agile methodology is a **flexible, iterative approach to software development** that focuses on delivering value to customers quickly and efficiently. Unlike traditional methods like the Waterfall model, Agile emphasizes adaptability, collaboration, and continuous improvement.

2. What is Agile?

- Agile is a **mindset** and a set of principles designed to handle projects where requirements are likely to change.
- It values:
    1. **Individuals and interactions** over processes and tools.
    2. **Working software** over comprehensive documentation.
    3. **Customer collaboration** over contract negotiation.
    4. **Responding to change** over following a plan.

This philosophy is outlined in the **Agile Manifesto**, created in 2001.

3. Core Principles of Agile

Agile has 12 guiding principles. Here are the most important ones explained simply:

1. **Customer Satisfaction:** Deliver valuable software quickly and regularly.
2. **Welcome Changes:** Embrace changes even late in development.
3. **Frequent Delivery:** Deliver working software in short cycles (2–4 weeks).
4. **Collaboration:** Developers, testers, and business teams work together daily.
5. **Motivated Teams:** Build projects around motivated individuals and trust them.
6. **Face-to-Face Communication:** Prefer direct communication for better understanding.
7. **Working Software is the Measure of Progress:** Focus on delivering usable products.
8. **Sustainable Development:** Maintain a constant pace of development without overworking.
9. **Technical Excellence:** Focus on high-quality work and good design.
10. **Simplicity:** Keep things as simple as possible.
11. **Self-Organizing Teams:** Let teams decide how to work for maximum efficiency.
12. **Regular Reflection:** Periodically review and adjust for continuous improvement.

4. Agile Frameworks

Agile is a broad concept implemented through different frameworks or methodologies. Here are the most popular ones:

4.1 Scrum Framework

Scrum is a widely adopted Agile framework focused on iterative development. It's designed to help teams deliver working software incrementally, responding to changes quickly. Scrum thrives on transparency, inspection, and adaptation.

A) Core Scrum Terminologies

1. **Product Backlog:**
    - A dynamic list of all desired product features, improvements, bugs, and technical tasks.
    - Managed by the **Product Owner** and continuously refined through **backlog grooming/refinement sessions**.
2. **Sprint Backlog:**
    - A subset of the Product Backlog, selected by the team during **Sprint Planning**.
    - Represents tasks committed to be delivered in the sprint.

3. **User Stories:**
   - Simple, concise descriptions of a feature or requirement written from the end-user's perspective.
   - Format: *As a [user role], I want [requirement] so that [business value].*
4. **Epics:**
   - Large user stories that can be broken down into smaller, actionable **user stories**.
5. **Story Points:**
   - A relative measure of effort required to complete a user story, typically estimated in Fibonacci sequence (e.g., 1, 2, 3, 5, 8, etc.).
6. **Sprint:**
   - A fixed-length iteration (typically 2–4 weeks) where the team delivers a potentially shippable product increment.
7. **Increment:**
   - The cumulative output of all completed tasks and stories in a sprint, delivered in a usable state.
8. **Burn-Down Chart:**
   - A visual representation of the amount of work remaining in the sprint over time.
9. **Burn-Up Chart:**
   - A graph showing work completed versus total scope, helping visualize progress and scope changes.
10. **Velocity:**
    - The average amount of work a team completes in a sprint, measured in story points or tasks.
11. **Definition of Ready (DoR):**
    - Criteria that a user story must meet before it is accepted into a sprint.
12. **Definition of Done (DoD):**
    - A checklist that ensures tasks are completed to a standard (e.g., code reviewed, tested, integrated, documented).

B) Scrum Roles

1. **Product Owner (PO):**
   - Owns and prioritizes the Product Backlog.
   - Acts as a bridge between stakeholders and the team.
2. **Scrum Master:**
   - Facilitates Scrum ceremonies.
   - Coaches the team in Scrum practices.
   - Removes impediments and shields the team from outside distractions.
3. **Development Team:**
   - Cross-functional and self-organizing group responsible for delivering the sprint backlog.
   - Includes developers, testers, designers, etc.

C) Artifacts in Scrum

1. **Product Backlog:**
   - Continuously refined during **Backlog Refinement** sessions.
   - Features are described in terms of **user stories**, **acceptance criteria**, and **dependencies**.
2. **Sprint Backlog:**
   - Includes detailed tasks for each selected story.
   - Often tracked visually using **Kanban boards** or similar tools.
3. **Increment:**
   - A usable portion of the product delivered after each sprint.

4. **Burndown/Burnup Charts:**
   - Track sprint and project progress.

D) Ceremonies in Scrum

1. **Sprint Planning:**
   - Held at the start of each sprint.
   - Objectives:
     - Define the sprint goal.
     - Select user stories for the sprint backlog.
     - Break down stories into actionable tasks.
2. **Daily Scrum (Stand-Up):**
   - A 15-minute meeting held every day of the sprint.
   - Each team member answers:
     - What did I do yesterday?
     - What will I do today?
     - Are there any blockers?
3. **Sprint Review:**
   - Held at the end of the sprint.
   - The team demonstrates the completed increment to stakeholders for feedback.
4. **Sprint Retrospective:**
   - A meeting to reflect on what went well, what didn't, and how to improve in the next sprint.
5. **Backlog Refinement (Grooming):**
   - Ongoing meeting to refine and prioritize the Product Backlog.

E) Tasks and Practices

1. **Estimations:** Use methods like **Planning Poker** to estimate user stories in story points.
2. **Task Breakdown:** Decompose user stories into smaller, actionable tasks.
3. **Continuous Testing:** Integrate automated and manual testing in the sprint cycle.

F) When to Use Scrum

- Projects with rapidly changing requirements.
- Teams with 5–9 members.
- Development environments needing frequent feedback and collaboration.

4.2 Kanban Framework

Kanban is a visual framework for managing workflow and reducing inefficiencies. Unlike Scrum, it does not have fixed-length iterations.

A) Core Terminologies

1. **Kanban Board:** A visual tool to track tasks across workflow stages (e.g., Backlog → To Do → In Progress → Done).
2. **Work in Progress (WIP) Limits:** The maximum number of tasks allowed in each stage to prevent overloading.
3. **Cycle Time:** The time a task takes to move from start to completion.
4. **Lead Time:** The time from task creation to its completion.

B) Artifacts in Kanban

1. **Kanban Board:** Columns represent workflow stages.
2. **Cards:** Represent tasks or work items, often color-coded by type (e.g., bug, feature, etc.).

C) Practices

1. **Pull System:** Team members pull tasks only when they have capacity.
2. **Continuous Flow:** Tasks move through stages without fixed-length sprints.
3. **Feedback Loops:** Regular check-ins to identify bottlenecks.

D) When to Use Kanban

- Teams handling ongoing work (e.g., support, operations).
- Projects with unpredictable workloads.

4.3 Extreme Programming (XP)

XP focuses on engineering practices to ensure high-quality code and rapid adaptability.

A) Core Practices

1. **Pair Programming:** Two developers write code together, improving quality.
2. **Test-Driven Development (TDD):** Writing tests before the actual code.
3. **Continuous Integration (CI):** Merge code changes frequently into a shared repository.
4. **Refactoring:** Continuously improving the codebase without changing functionality.
5. **Small Releases:** Frequent, incremental deliveries.

B) Terminologies

1. **Spike:** A time-boxed task to explore solutions or reduce uncertainty.
2. **User Stories:** Captures what the customer needs.

4.4 Scaled Agile Framework (SAFe)

SAFe (Scaled Agile Framework) is designed for large organizations to apply Agile practices across multiple teams. It emphasizes alignment, collaboration, and delivery at scale through a hierarchical structure of teams, programs, and portfolios.

A) Core Terminologies

1. **Agile Release Train (ART):** A long-lived team of multiple Agile teams (50–125 members) that work together to deliver value incrementally.
2. **Program Increment (PI):** A time-boxed period (8–12 weeks) consisting of multiple sprints, during which teams deliver value.
3. **Portfolio Backlog:** High-level list of business epics, initiatives, and features managed at the portfolio level.
4. **Solution Intent:** A shared understanding of what a solution must do, including functional and non-functional requirements.

B) Roles

1. **Release Train Engineer (RTE):** Scrum Master at the ART level, ensuring teams stay aligned and remove impediments.
2. **Product Management:** Works at the program level to prioritize and refine features.
3. **System Architect:** Designs the technical architecture for the solution.
4. **Epic Owner:** Manages the delivery of epics, which are large cross-cutting initiatives.

C) Artifacts

1. **Program Backlog:** Features planned for delivery by ARTs.
2. **Solution Backlog:** Large-scale initiatives requiring multiple ARTs.
3. **PI Objectives:** Team and ART-level objectives for a Program Increment.

D) Ceremonies

1. **PI Planning:** A two-day event where all teams in an ART collaborate to plan the objectives and dependencies for the next PI.
2. **System Demo:** A demonstration of the integrated work across all teams at the end of each sprint.
3. **Inspect and Adapt Workshop:** A session to reflect on the ART's performance after a PI and plan improvements.

E) When to Use SAFe

- Large organizations with multiple Agile teams.
- Projects requiring alignment across departments and portfolios.

4.5 Lean Development

Lean Development is inspired by Lean Manufacturing principles, focusing on delivering value and minimizing waste.

A) Core Terminologies

1. **Value Stream Mapping:** A tool to visualize and analyze the flow of materials and information in delivering a product.
2. **Muda (Waste):** Anything that doesn't add value to the customer, categorized as defects, overproduction, waiting, and more.
3. **Just-In-Time (JIT):** Delivering only what is needed when it's needed.

B) Principles

1. **Eliminate Waste:** Remove activities that don't add value.
2. **Amplify Learning:** Use continuous feedback loops.
3. **Deliver as Fast as Possible:** Optimize flow to reduce delays.
4. **Decide Late:** Make decisions based on the latest possible data.

C) Practices

- **Pull System:** Tasks are only started when there's a demand.
- **Kaizen:** Continuous improvement in processes.

D) When to Use Lean

- Projects requiring rapid iteration with a focus on efficiency.
- Scenarios with a high degree of uncertainty or evolving requirements.

4.6 Dynamic Systems Development Method (DSDM)

DSDM is a project management and delivery framework focused on delivering business value early and iteratively. It combines Agile principles with a strong governance structure.

A) Core Terminologies

1. **MoSCoW Prioritization:**
    - Must have, Should have, Could have, Won't have.
    - Ensures focus on delivering essential requirements.
2. **Timeboxing:**
    - Fixed delivery periods with predefined deliverables.

B) Roles

1. **Business Sponsor:** Responsible for funding and ensuring alignment with business objectives.
2. **Technical Coordinator:** Ensures the technical viability of the project.
3. **Solution Developer and Tester:** Builds and verifies the solution iteratively.

C) Artifacts

1. **Feasibility Report:** Outlines whether the project is viable.
2. **Solution Backlog:** A prioritized list of features and technical work.

D) Ceremonies

- Iteration Planning and Review.
- Daily Standups.
- Post-Project Reviews to assess outcomes.

E) When to Use DSDM

- Projects requiring strict governance.
- Scenarios with fixed deadlines and budgets.

4.7 Crystal Methodology

Crystal focuses on people, communication, and adapting the process to the team's needs. It has several variations (Crystal Clear, Crystal Orange) based on team size and criticality.

A) Core Terminologies

1. **Core Roles:** Sponsor, Lead Designer, and Developer.
2. **Incremental Development:** Delivering usable increments frequently.

C) Practices

1. **Frequent Delivery:** Ensure a working product is delivered often.
2. **Reflection Workshops:** Regular retrospectives for improvement.

D) When to Use Crystal

- Small teams with less critical projects.
- High emphasis on communication and individual strengths.

5. Comparative Table: Agile Frameworks

Aspect	Scrum	Kanban	(XP)	SAFe	Lean	DSDM	Crystal
Primary Focus	Iterative delivery with structured roles & events.	Continuous flow of tasks with WIP limits.	Engineering excellence & rapid releases.	Scaling Agile for large enterprises.	Reducing waste & maximizing value.	Governance-focused iterative delivery.	Team-specific, communication-driven process.
Iteration Cycle	Time-boxed sprints (2–4 weeks).	Continuous (no fixed iterations).	Iterations of 1–2 weeks.	Program Increment (8–12 weeks) with sprints.	Continuous flow (adaptive iterations).	Timeboxed iterations.	Flexible, as per team needs.
Key Roles	Product Owner, Scrum Master, Dev Team.	None (team self-organizes).	Coach, Customer, Developers.	RTE, Product Manager, System Architect.	Team roles vary, focus on value streams.	Sponsor, Coordinator, Solution Devs.	Sponsor, Lead Designer, Developers.
Prioritization	Product & Sprint Backlogs	Kanban board prioritization	User stories with customer-driven prioritization.	Portfolio Backlog, Program Backlog.	Value Stream Mapping, JIT prioritization.	MoSCoW: Must/Should/Could/Won't Have.	Informal backlog management.
Artifacts	Backlogs, Increment, Burn Charts.	Kanban Board, Work Items.	Codebase, User Stories, Test Cases.	Program Backlog, PI Objectives, Solution Intent.	Value Stream Map, Kaizen Events.	Feasibility Report, Solution Backlog.	Lightweight or customized.

Ceremonies	Sprint Planning, Daily Scrum, Review, Retro.	Optional: Standups, Feedback Loops.	Planning Game, Standups, Retrospectives.	PI Planning, System Demos, Inspect & Adapt.	Flow optimization workshops.	Iteration Planning, Post-Project Reviews.	Reflection Workshops, Informal Reviews.
Practices	Timeboxing, Velocity, Estimation.	Pull system, Continuous Delivery.	Pair Programming, TDD, Continuous Integration.	Alignment across teams, Agile Release Trains.	Eliminate Waste, Amplify Learning.	Strong focus on documentation & planning.	Focus on team culture & communication.
Best For	(5–9 members) working on evolving products.	Maintenance/Support teams with ongoing work.	Small teams needing technical excellence.	Large organizations with multiple Agile teams.	Teams focused on efficiency & lean processes.	Projects with strict governance & time constraints.	Teams with strong communication & adaptability.