PROJECT MANAGEMENT

Unit II – Agile Software Development

2.1 Introduction to Agile Software Development

In traditional software development, teams used to plan everything at the start of a project and then follow that plan very strictly. This approach worked well for simple and predictable projects, but it had problems when requirements kept changing. Agile software development was created to solve these problems.

> **Definition (Agile Manifesto, 2001)**
> *"Agile is a set of values and principles that guide software development through iterative delivery, collaboration, and adaptability to change."*

The word "Agile" means quick and flexible. In software development, Agile means building software in small, working pieces step-by-step, getting feedback from the customer at every step, and making changes quickly when needed.

The Agile Manifesto – Four Core Values

The Agile Manifesto was written in 2001 by 17 software experts. It defines four important values:

- **Individuals and interactions** over processes and tools. (People and teamwork are more important than rigid processes.)
- **Working software** over comprehensive documentation. (A working product is more valuable than lots of documents.)
- **Customer collaboration** over contract negotiation. (Working with the customer is more important than following a fixed contract.)
- **Responding to change** over following a plan. (Being flexible and adapting to change is better than blindly following a plan.)

Twelve Principles of Agile

The Agile Manifesto also lists twelve guiding principles. The most important ones are:

- Deliver working software frequently (every few weeks).
- Welcome changing requirements, even late in development.
- Business people and developers must work together daily.
- Build projects around motivated individuals and trust them.
- Working software is the primary measure of progress.
- Sustainable development – maintain a constant pace indefinitely.
- Continuous attention to technical excellence.
- Simplicity – maximize the amount of work not done.

2.2 Agile Methods

Agile is not a single method. It is a family of approaches that all follow the Agile values and principles. Different organisations and projects may choose different Agile methods based on their needs.

The most widely used Agile methods are:

1. Scrum

Scrum is the most popular Agile framework. It divides the project into fixed time periods called Sprints (usually 2–4 weeks). After each Sprint, a working piece of software is delivered to the customer.

2. Extreme Programming (XP)

XP focuses on improving software quality and responding to changing customer requirements. It uses practices like Test-Driven Development (TDD), pair programming, and continuous integration.

3. Kanban

Kanban is a visual method that uses a Kanban board with columns like "To Do", "In Progress", and "Done" to manage and visualise work. It does not use fixed time iterations.

4. Feature Driven Development (FDD)

FDD focuses on building features one by one. The team creates a list of all features, then designs and builds them in short two-week cycles.

5. Dynamic Systems Development Method (DSDM)

DSDM is a framework that focuses on delivering the right solution at the right time. It emphasises business needs and uses timeboxing (fixed time limits for each activity).

6. Crystal Methods

Crystal is a family of methods (Crystal Clear, Crystal Yellow, Crystal Orange) that are chosen based on the size and criticality of the project. Smaller projects use lighter methods.

2.3 Scrum

Scrum is the most widely used Agile framework in the software industry. It helps teams deliver working software in short, fixed-length periods called Sprints.

> **Definition**
> *"Scrum is an Agile framework for managing and completing complex projects. It uses iterative Sprints, defined roles, and regular meetings to deliver value to the customer continuously."*

Key Concepts in Scrum

Sprint

A Sprint is a fixed time period (usually 2 to 4 weeks) during which the team works to complete a set of tasks from the backlog. At the end of each Sprint, a working, potentially shippable product is produced.

Product Backlog

The Product Backlog is a prioritised list of all features, requirements, and tasks that need to be done in the project. It is maintained by the Product Owner and is regularly updated.

Sprint Backlog

The Sprint Backlog is the list of tasks the team commits to completing in the current Sprint. It is created at the beginning of each Sprint during the Sprint Planning meeting.

Scrum Roles

- Product Owner – Represents the customer, prioritises the Product Backlog, defines user stories.
- Scrum Master – Facilitates the Scrum process, removes obstacles for the team, ensures Scrum practices are followed.
- Development Team – Cross-functional team that designs, codes, and tests the product. Usually 3–10 members.

Scrum Events (Meetings)

- **Sprint Planning:** Held at the start of each Sprint. Team decides what to work on in the Sprint and how to do it.
- **Daily Scrum (Stand-up):** A short 15-minute meeting every day. Each team member answers: What did I do yesterday? What will I do today? Are there any blockers?
- **Sprint Review:** Held at the end of each Sprint. Team demonstrates the completed work to stakeholders and gets feedback.
- **Sprint Retrospective:** Team reflects on the Sprint process to find ways to improve. Held after the Sprint Review.

Scrum Artefacts

- **Product Backlog:** List of all work to be done in the project.
- **Sprint Backlog:** Work selected for the current Sprint.
- **Increment:** The working, potentially shippable product at the end of each Sprint.
- **Burndown Chart:** A visual chart showing work remaining in the Sprint or project over time.

2.4 Comparison between Non-Agile and Agile Project

Traditional (Non-Agile) project management follows a plan-driven, sequential approach. Agile project management is flexible and iterative. The table below clearly shows the key differences.

Feature	Non-Agile (Waterfall)	Agile
Approach	Sequential, phase by phase	Iterative and incremental
Planning	All planning done at the start	Planning done continuously
Requirements	Fixed at the beginning	Can change throughout the project
Customer Involvement	Only at start and end	Continuous throughout the project
Delivery	Single delivery at the end	Frequent deliveries every Sprint
Testing	Done after development	Done continuously throughout
Change Management	Changes are difficult and costly	Changes are welcomed and easy to adopt
Documentation	Heavy documentation required	Minimal, just enough documentation
Team Structure	Hierarchical, role-based	Self-organising, cross-functional teams
Risk Management	Risks identified early, hard to adjust	Risks addressed continuously each Sprint
Best Suited For	Fixed, well-understood requirements	Complex, evolving requirements

2.5 Three Stages of an Agile Project

Although Agile is flexible, every Agile project generally moves through three broad stages. These stages are not as rigid as the Waterfall phases, but they help organise the work.

Stage 1: Envision (Project Initiation)

This is the starting stage of the Agile project. The team and the customer come together to define the big picture of the project.

- Identify the product vision and the main goal of the project.
- Create the initial Product Backlog with all known requirements.
- Form the Agile team (Product Owner, Scrum Master, Development Team).
- Decide on the Agile method (Scrum, XP, Kanban, etc.) to be used.
- Set the overall timeline and budget.

The main output of this stage is the Product Vision and the initial Product Backlog.

Stage 2: Speculate and Explore (Iterative Development)

This is the main working stage. The team works in iterations (Sprints) to build and deliver the product. Each iteration includes planning, development, testing, and review.

- Sprint Planning: Select items from the Product Backlog for the Sprint.
- Design, develop, and test the selected features.
- Daily Scrum meetings to track progress and solve problems.
- Sprint Review: Demonstrate the working product to the customer.
- Sprint Retrospective: Improve team processes.
- Update the Product Backlog based on customer feedback.

This stage continues for multiple Sprints until all major features are delivered.

Stage 3: Close (Project Closure)

This is the final stage of the Agile project. The team completes all remaining tasks and delivers the final product to the customer.

- Complete any remaining items from the Product Backlog.
- Final testing and quality assurance.
- Customer acceptance and sign-off.
- Deploy the final product to the production environment.
- Conduct a final retrospective and document lessons learned.
- Release the project team.

2.6 Plan-Driven and Agile Development

Plan-driven development and Agile development represent two very different philosophies of software development. Many organisations today use a hybrid approach that combines elements of both.

Plan-Driven Development

In plan-driven development, everything is planned in detail at the beginning of the project. The team strictly follows the plan throughout the project lifecycle.

- **Predictability:** The scope, cost, and timeline are defined at the start, making it easy to predict outcomes.
- **Documentation:** Heavy documentation is created at every stage.
- **Sequential Process:** Each phase (Requirements → Design → Development → Testing → Deployment) must be completed before the next begins.
- **Low Flexibility:** Changes to requirements are difficult and expensive once development begins.
- **Best For:** Projects with stable, well-defined requirements such as construction, aerospace, or government projects.

Agile Development

In Agile development, the team works in small iterations, continuously delivering working software and adapting to changing requirements.

- **Flexibility:** Requirements can change at any time, and the team adapts quickly.
- **Continuous Delivery:** Working software is delivered after every Sprint.
- **Customer Collaboration:** The customer is involved throughout the project, providing feedback regularly.
- **Minimal Documentation:** Only essential documentation is created.
- **Best For:** Projects with evolving requirements, such as web applications, mobile apps, and startup products.

Choosing Between Plan-Driven and Agile

The choice between Plan-Driven and Agile depends on several factors:

- If requirements are clear and unlikely to change → Use Plan-Driven.
- If requirements are unclear or likely to change → Use Agile.
- If the team is large and distributed → Plan-Driven may work better.
- If the team is small and co-located → Agile works best.
- For safety-critical systems (medical, aviation) → Plan-Driven with strict documentation.

2.7 Extreme Programming (XP)

Extreme Programming (XP) is one of the earliest and most well-known Agile methods. It was developed by Kent Beck in the late 1990s. XP focuses on producing high-quality software and improving the quality of the development process itself.

> **Definition**
> *"Extreme Programming (XP) is an Agile software development methodology that aims to improve software quality and responsiveness to changing customer requirements through specific engineering practices and values."*

Five Values of XP

- **Communication:** All team members must communicate openly and frequently with each other and with the customer.
- **Simplicity:** Always do the simplest thing that could possibly work. Avoid unnecessary complexity.
- **Feedback:** Get feedback from the customer frequently to make sure the right product is being built.
- **Courage:** Have the courage to make difficult decisions, to change the design, or to throw away bad code.

- **Respect:** Every team member must respect each other and value each other's contributions.

Key XP Practices

- **Test-Driven Development (TDD):** Write the test first, then write the code to pass the test. This ensures code quality from the very beginning.
- **Pair Programming:** Two developers work together on the same code at the same computer. One writes code, the other reviews. This reduces errors and improves code quality.
- **Continuous Integration:** Code is integrated and tested multiple times a day to detect errors early.
- **Refactoring:** Continuously improve the internal structure of the code without changing its behaviour.
- **Small Releases:** Release small, working versions of the software very frequently (sometimes daily).
- **On-Site Customer:** A representative of the customer works directly with the development team throughout the project.
- **Collective Code Ownership:** Any developer can modify any part of the code. No one "owns" a specific module.
- **Coding Standards:** All developers follow the same coding conventions so that any developer can read and understand any code.
- **Sustainable Pace:** The team should not work overtime. A sustainable 40-hour workweek keeps quality high.
- **Planning Game:** The customer writes user stories (short descriptions of features); the team estimates effort; then both agree on what goes into the next release.

XP Release and Iteration Plan

XP uses two levels of planning:

- **Release Plan:** Covers 3–6 months. Defines which user stories will be delivered in each iteration.
- **Iteration Plan:** Covers 1–3 weeks. Breaks user stories into specific development tasks.

2.8 Scaling Agile Methods

Agile methods were originally designed for small teams (5–10 people). However, large organisations need Agile to work with hundreds or thousands of developers across multiple teams and locations. This is called Scaling Agile.

Challenges in Scaling Agile

- Coordinating multiple Agile teams working on the same product.
- Ensuring consistent communication across large, distributed teams.
- Aligning multiple teams to the same product vision and goals.

- Managing dependencies between different teams' work.
- Integrating work from multiple teams into one coherent product.

Popular Frameworks for Scaling Agile

1. SAFe – Scaled Agile Framework

SAFe is the most widely used framework for large-scale Agile. It organises multiple Agile teams into Agile Release Trains (ARTs) – groups of teams that work together towards a common Programme Increment (PI), usually lasting 8–12 weeks. SAFe has four levels: Team, Programme, Large Solution, and Portfolio.

2. LeSS – Large-Scale Scrum

LeSS is a simple extension of Scrum for multiple teams. It uses a single Product Backlog, a single Product Owner, and one overall Sprint across all teams. It emphasises keeping Scrum simple and avoiding unnecessary processes.

3. DAD – Disciplined Agile Delivery

DAD is a hybrid framework that combines Scrum, XP, Lean, and other methods. It provides a toolkit of practices and allows teams to choose the approach that best fits their context.

4. Nexus

Nexus is an extension of Scrum developed by Scrum.org for scaling Scrum to 3–9 teams. It adds a Nexus Integration Team that coordinates work across all teams and manages integration issues.

Scrum of Scrums

Scrum of Scrums is a simple coordination technique where one representative from each Scrum team meets regularly (usually daily or weekly) to synchronise work, identify dependencies, and resolve cross-team issues.

2.9 Roles and Responsibilities in Agile

Agile teams have clearly defined roles with specific responsibilities. Each role is essential for the success of the project. Unlike traditional projects, Agile teams are self-organising and cross-functional.

Product Owner

The Product Owner (PO) represents the customer and is responsible for maximising the value of the product.

- Defines and prioritises the Product Backlog.
- Writes User Stories (descriptions of features from the customer's point of view).
- Accepts or rejects work at the Sprint Review.
- Communicates the product vision to the team.

- Serves as the single point of contact for all customer requirements.

Scrum Master

The Scrum Master is the servant-leader of the Scrum team. They ensure that the team follows the Scrum framework and is able to work effectively.

- Facilitates all Scrum events (Sprint Planning, Daily Scrum, Sprint Review, Retrospective).
- Removes obstacles (called impediments) that block the team.
- Coaches the team on Agile and Scrum practices.
- Protects the team from external distractions and interference.
- Helps the organisation adopt Agile practices.

Development Team

The Development Team is the group of professionals who do the actual work of designing, coding, and testing the software.

- Self-organising: the team decides how to do the work.
- Cross-functional: the team has all the skills needed (design, development, testing).
- Collectively responsible for the quality of each Sprint output.
- Estimates the effort for backlog items.
- Participates in all Scrum events.

Stakeholders

Stakeholders are anyone who has an interest in the project – customers, managers, end users, investors. In Agile, stakeholders are actively involved, especially during Sprint Reviews, providing feedback that guides the direction of the product.

2.10 Scheduling and Tracking in Agile

In Agile, scheduling and tracking are done differently compared to traditional methods. Rather than a fixed, detailed plan for the entire project, Agile uses rolling wave planning – detailed planning is done for the near future and lighter planning for later work.

Agile Scheduling

Agile scheduling is done at multiple levels:

- **Release Planning:** Defines which features will be delivered in each release over the next 3–6 months. Based on the velocity (amount of work a team completes per Sprint).
- **Sprint Planning:** Decides what work the team will do in the upcoming Sprint (1–4 weeks).

- **Daily Planning:** Each day, the team adjusts their work in the Daily Scrum based on progress and obstacles.

Velocity

Velocity is the amount of work (measured in Story Points) that a team completes in one Sprint. Velocity is used to predict how much work can be done in future Sprints and to create the release schedule.

Story Points

Story Points are a unit of measurement used to estimate the effort required to complete a user story. They consider complexity, uncertainty, and effort – not just time. Teams use Planning Poker to estimate story points collaboratively.

Agile Tracking Tools

- **Burndown Chart:** Shows how much work remains in the Sprint or Release. The ideal burndown is a straight diagonal line from total work to zero. If the actual line is above the ideal, the team is behind schedule.
- **Burnup Chart:** Shows how much work has been completed over time. Unlike the burndown, it also shows changes in the total scope of the project.
- **Kanban Board:** A visual board with columns (To Do → In Progress → Done) showing the status of each task in the Sprint.
- **Velocity Chart:** Shows the team's velocity (story points completed) across multiple Sprints to help predict future capacity.
- **Sprint Burndown:** A daily chart showing the remaining work in the current Sprint.

Tracking Progress in Agile – Key Practices

- Daily Scrum: Team members report progress and identify blockers every day.
- Sprint Review: Completed work is demonstrated at the end of each Sprint.
- Definition of Done (DoD): A checklist of criteria that must be met for a task to be considered "done" (e.g., coded, tested, reviewed, documented).
- Product Backlog Refinement: The Product Owner regularly reviews and updates the backlog to ensure priorities are current.