Summary of Lecture 4: Agile Software Development

This lecture covers **Agile Software Development**, its philosophy, principles, and different Agile methodologies. Agile was introduced as an alternative to **traditional "heavy" software processes** (e.g., Waterfall, Unified Process) to be more adaptable to **changing requirements** and **faster delivery cycles**.

What is Agile? Why "Go Agile"?

- Traditional software development models (e.g., Waterfall, UP) are **rigid** and **plan-heavy**.
- Agile was created to handle **uncertain and changing requirements** better.
- In **2001**, the **Agile Alliance** introduced the **Agile Manifesto**, which defines Agile values.

Agile Manifesto: Core Values

Agile emphasizes **flexibility, collaboration, and working software** over rigid processes.

1️⃣ **Individuals & interactions** over **processes & tools**

- Teams should be **self-organized and adaptable** rather than follow strict processes.

2️⃣ **Working software** over **comprehensive documentation**

- Deliver **actual software** frequently, rather than spending too much time on planning documents.

3️⃣ **Customer collaboration** over **contract negotiation**

- Clients and developers **work together continuously** instead of just negotiating requirements once.

4️⃣ **Responding to change** over **following a plan**

- Agile **embraces change**, even late in development, instead of sticking to a rigid plan.

Agile Levels: Key Components

1️⃣ **Agile Values** – The **philosophy** behind Agile (from the manifesto).
2️⃣ **Agile Principles** – **Strategies** that define how Agile is implemented.
3️⃣ **Agile Methods** – Specific **frameworks** (e.g., Scrum, XP, RAD).

4️⃣ **Agile Practices** – **Techniques** used within Agile (e.g., stand-up meetings, product backlogs).

Agile Principles (Key Ideas)

📌 **Deliver working software frequently** (every **1–6 weeks**).
📌 **Embrace change**, even late in development.
📌 **Close collaboration** between developers and business teams.
📌 **Face-to-face communication** is preferred over emails/docs.
📌 **Simplicity** – Do the **minimum work** required for success.
📌 **Self-organizing teams** make the best decisions.
📌 Regularly **reflect & adjust** for continuous improvement.

Agile Methods: How Agile is Implemented

Different Agile frameworks apply Agile principles in **different ways**.

Agile Manifesto Value	Agile Techniques
Individuals & interactions	Small teams, pair programming, shared code ownership
Working software	Test-driven development, refactoring
Customer collaboration	Customer representatives within the team
Responding to change	Continuous integration, frequent releases

---

Agile Development Cycle

Instead of a **fixed design phase**, Agile follows **iterative cycles**:

1️⃣ **Obtain requirements** (for a small part of the project).
2️⃣ **Modify code and test** (implementing new requirements).
3️⃣ **Refactor** (clean and improve the codebase).
4️⃣ Repeat the cycle for the next requirements.

💡 **Key difference from UP**: Agile **does not** have a separate, explicit **design phase**—instead, the design evolves through **refactoring** after each iteration.

Minimum Viable Product (MVP)

MVP = The smallest possible version of a product that delivers value.

- Focuses only on **core functionalities** necessary for the product to work.
- Anything that is **not essential** is removed to **deliver quickly**.

💡 **Example:**

- A **social media MVP** might only include **posting & viewing posts**, leaving out likes, comments, or profiles for later iterations.

Rapid Application Development (RAD)

RAD is an **Agile-like** approach that focuses on **speed and user involvement**.

- Created in **1991** by **James Martin**.
- Uses **prototyping**, **small development teams**, and **automated tools**.
- Works in **fixed time frames (Time Boxes)** instead of fixed scope.

RAD Lifecycle

1️⃣ **Requirement Planning** – Gather & prioritize requirements using **Joint Requirement Planning (JRP)** workshops.
2️⃣ **User Design** – Users work **directly with developers** to refine system design (**Joint Application Design - JAD**).
3️⃣ **Construction** – A **SWAT (Skilled With Advanced Tools) team** rapidly builds multiple prototypes.
4️⃣ **Cutover** – **Testing, deployment, and training** take place.

MoSCoW Prioritization (Used in RAD)

- **M**ust-have – Essential features for the current iteration.
- **S**hould-have – Important but not mandatory features.
- **C**ould-have – Nice-to-have but not necessary.
- **W**on't-have – Features postponed for future versions.

Dynamic Systems Development Method (DSDM)

DSDM is an **Agile method** that evolved from **RAD**.

- Uses **fixed time and resources** (Time Boxed like RAD).
- Prioritizes requirements using **MoSCoW**.
- **Iterative and incremental** like other Agile frameworks.

DSDM Stages

1️⃣ **Feasibility** – Outline plan, check if Agile is suitable.
2️⃣ **Business Study** – High-level analysis of business needs.
3️⃣ **Functional Model Iteration** – Produce analysis models and functioning prototypes.
4️⃣ **Design & Build Iteration** – Integrate functional components into a working system.
5️⃣ **Implementation** – System testing and deployment.

DSDM Best Practices

✅ **Active user involvement** throughout development.
✅ **Empowered teams** make their own decisions.
✅ **Frequent deliveries** of working software.
✅ **Testing is integrated** into the development lifecycle.

Key Differences: Agile vs. Unified Process (UP) vs. Waterfall

Feature	Waterfall	Unified Process (UP)	Agile
Flexibility	Rigid, plan-heavy	Somewhat flexible (phased)	Highly flexible
Iterations	None	Structured iterations per phase	Short, continuous iterations
User Involvement	Mostly in requirement phase	Regular feedback per phase	Continuous collaboration
Documentation	Extensive upfront docs	Moderate docs per phase	Minimal, just enough docs
Risk Management	Late risk discovery	Risk-driven, assessed per phase	Adaptable, risks handled continuously
Best For	Large, **stable projects**	Large, **structured projects**	**Fast-changing, dynamic projects**

Final Thoughts

🔹 **Agile is best for fast-moving, adaptable projects** where requirements change often.
🔹 **Unified Process (UP) is structured but flexible**, making it better for large, **complex systems**.
🔹 **Waterfall is only suitable for projects with stable and clear requirements** from the start.

If flexibility, collaboration, and speed matter → Agile is the best approach.

Keywords

- **Agile Software Development**
- **Agile Manifesto**
- **Agile Values & Principles**
- **Iterative & Incremental Development**
- **Minimum Viable Product (MVP)**
- **Rapid Application Development (RAD)**
- **MoSCoW Prioritization**
- **Dynamic Systems Development Method (DSDM)**
- **Time Boxing**
- **Refactoring**
- **Joint Requirement Planning (JRP)**
- **Joint Application Design (JAD)**
- **Self-organizing Teams**