AGILE SOFTWARE DEVELOPMENT (3 HRS.)

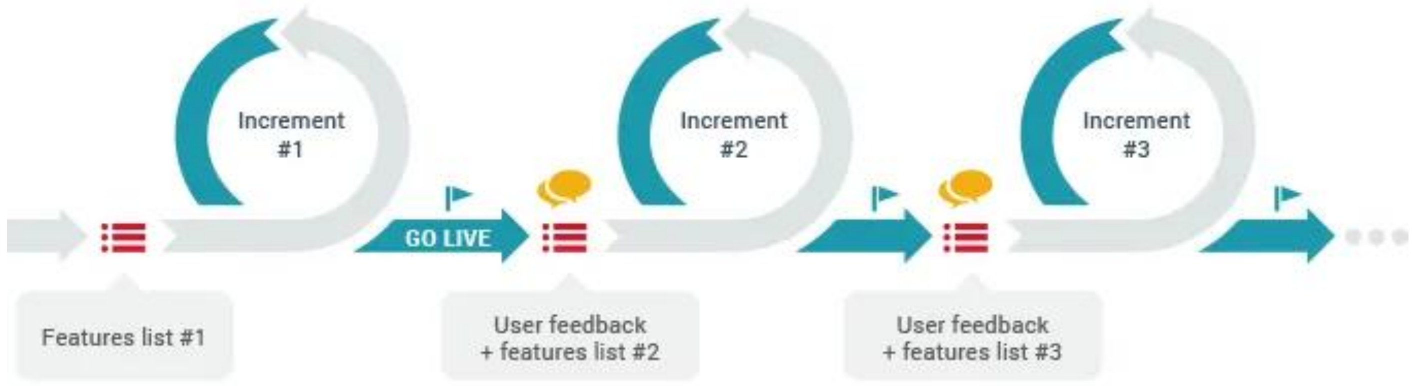
Unit 3

Prepared by: Natabar Khatri

New Summit College

AGILE DEVELOPMENT

- Agile methods emerged in the late 1990s as a response to the limitations of traditional, plan-driven software development approaches.
- The term agile refers to **being able to move quickly and easily**. It also refers **flexibility and the willingness and ability to change and adapt**.
- **The Need for Agile Development**
- In today's fast-paced business environment, organizations face:
  - **Global competition** requiring rapid response to market changes
  - **Evolving customer needs** that make fixed requirements impractical
  - **Pressure to deploy software quickly**, often prioritizing speed over perfection


Increment #1
Increment #2
Increment #3
GO LIVE
Features list #1
User feedback + features list #2
User feedback + features list #3

AGILE DEVELOPMENT

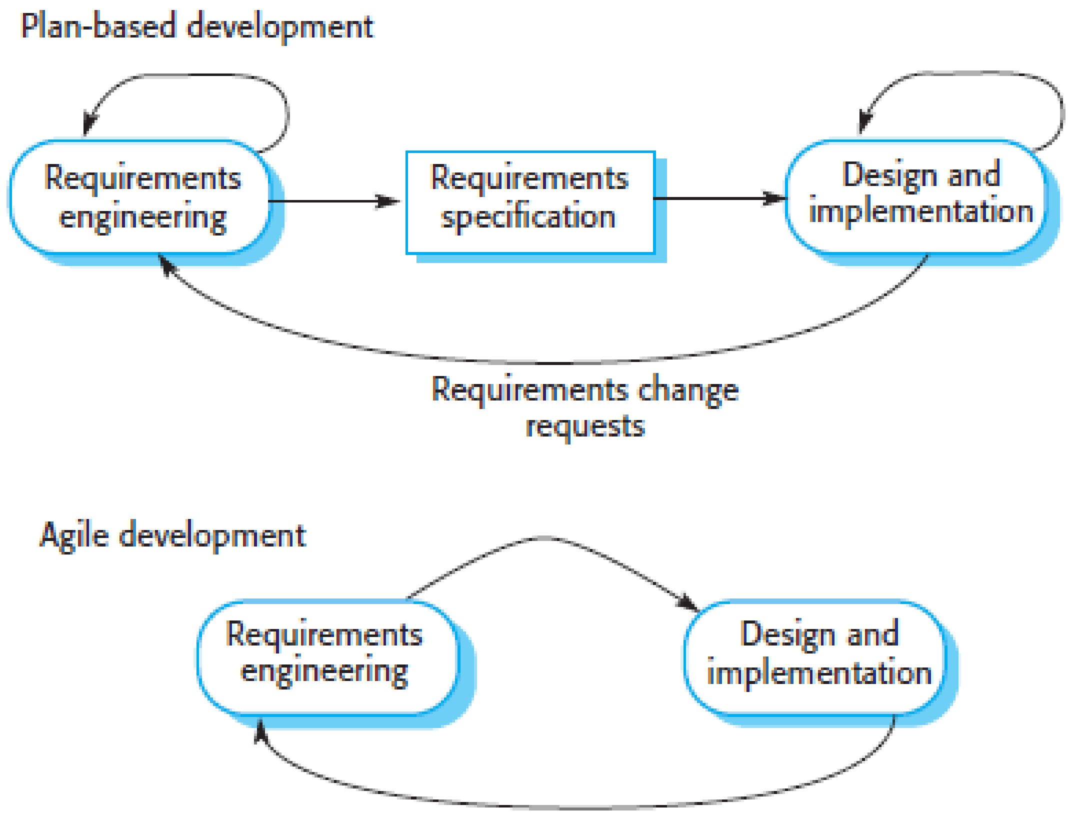
Common Characteristics of Agile Development

- **Interleaved Activities**
  - Specification, design, and implementation occur simultaneously
  - Minimal upfront documentation (focus on working software)
  - The user requirements document is an outline definition of the most important characteristics of the system.
- **Incremental Delivery**
  - Small, frequent releases (typically every 2-3 weeks)
  - Stakeholders evaluate and refine requirements iteratively
- **Customer Collaboration**
  - End-users actively participate throughout development
  - Feedback directly shapes future iterations

Extensive tool support is used to support the development process. Tools that may be used include automated testing tools, tools to support configuration management, and system integration and tools to automate user interface production.

PLAN-DRIVEN VS. AGILE DEVELOPMENT

- In plan-driven methods (e.g., Waterfall, V-Model), iteration occurs **within individual stages** of the process, with formal documents acting as handoffs between phases.
- Agile methods (e.g., Scrum, XP) blend requirements, design, and implementation into **cross-functional iterations** (sprints).


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

PLAN-DRIVEN VS. AGILE DEVELOPMENT

Plan-Driven Process Structure

- Linear phase execution:
  - Requirements → Design → Implementation → Testing → Maintenance
- Iteration occurs within phases
- Formal documents transition between stages
- Changes trigger phase rework

Agile Process Structure

- Cyclic sprints (typically 2-4 weeks)
- Cross-functional iteration:
  - Requirements refinement
  - Design adjustment
  - Implementation
  - Testing
- Continuous integration
- Regular stakeholder feedback

PLAN-DRIVEN VS. AGILE DEVELOPMENT

When to Use Each

- **Plan-Driven excels when:**
  - Requirements are stable and well-understood
  - Regulatory compliance demands documentation • Large-scale systems integration required
  - High consequence of failure (e.g., medical devices)
- **Agile excels when:**
  - Requirements are uncertain or volatile
  - Rapid time-to-market is critical
  - Customer feedback is essential
  - Innovation and experimentation needed

AGILE METHODS

Why Agile Methods Were Developed

- **Dissatisfaction with Heavyweight Processes**
  - Traditional plan-driven methods (e.g., Waterfall) required extensive upfront planning, detailed documentation, and strict phase completion before moving forward.
  - This led to **slow development cycles**, high overhead costs, and difficulty adapting to requirement changes.
- **Need for Faster, More Responsive Development**
  - Businesses demanded **quicker software releases** to stay competitive.
  - Agile methods enabled **incremental delivery**, allowing teams to release functional software in weeks rather than years.

AGILE METHODS

- **Shift in Development Priorities**
- The Agile Manifesto (2001) redefined success by valuing:
  - **Individuals and interactions** over processes and tools
  - **Working software** over comprehensive documentation.
  - **Customer collaboration** over rigid contract negotiation.
  - **Responding to change** over following a fixed plan.

AGILE METHODS

- **The Agile Process in Practice**

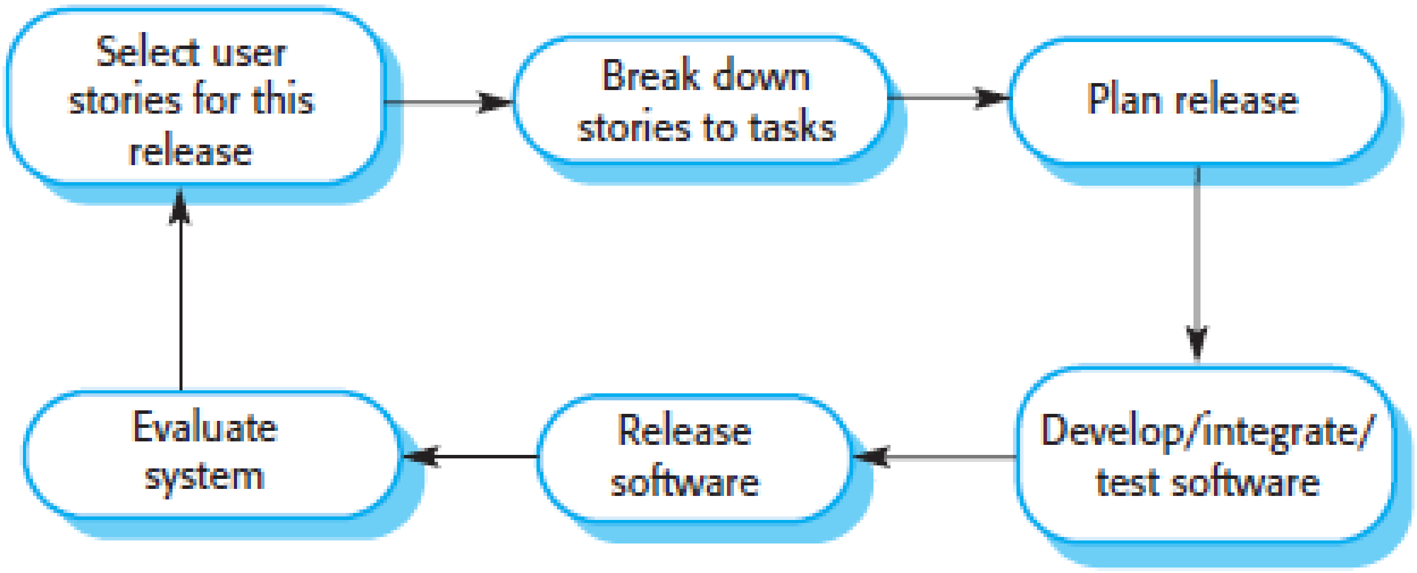

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Fig: The XP release cycle

AGILE METHODS

- **The Agile Process in Practice**
- **Select User Stories**
  - Prioritize features for the next release (e.g., "As a user, I want to reset my password").
- **Break Down Tasks**
  - Divide stories into actionable items (e.g., design UI, implement backend logic).
- **Plan the Release**
  - Define sprint goals and timelines.
- **Develop, Integrate & Test**
  - Code, merge changes, and run automated tests daily.
- **Evaluate & Release**
  - Demo to stakeholders, gather feedback, and deploy.

Example: A team building an e-commerce site:

- **Sprint 1:** User login functionality
- **Sprint 2:** Product search feature
- **Sprint 3:** Shopping cart integration

AGILE DEVELOPMENT TECHNIQUES

- Agile development revolutionized software engineering by shifting focus from rigid, documentation-heavy processes to flexible, iterative, and collaborative methods.
- Among these, **Extreme Programming (XP)** emerged as one of the most influential frameworks, pushing agile principles to their limits to maximize efficiency and adaptability.

Core Principles of Extreme Programming (XP)

XP is built on practices that emphasize **rapid delivery, continuous feedback, and high code quality**. Extreme programming embraces the idea of pair programming and test-driven development.

1. **Incremental Development via Small Releases**
2. **Continuous Customer Involvement**
3. **People-Centric Development**
4. **Embracing Change**
5. **Simplicity in Design**

AGILE DEVELOPMENT TECHNIQUES

1. Incremental Development via Small Releases

- **User Stories:** Requirements are captured as simple, customer-centric scenarios (e.g., "As a user, I want to reset my password").
- **Frequent Releases:** Instead of waiting months for a final product, XP teams deliver **working increments in days or weeks**.
- **Prioritization:** Features are selected based on business value and feasibility, ensuring the most critical functions are developed first.

Example: A team building an e-commerce site releases a basic product listing first, then iteratively adds a shopping cart, checkout, and payment processing.

AGILE DEVELOPMENT TECHNIQUES

2. Continuous Customer Involvement

- **On-Site Customer:** A dedicated customer representative works **full-time** with the development team to clarify requirements and validate features.
- **Acceptance Testing:** The customer defines test cases to ensure the software meets real-world needs.

3. People-Centric Development

- **Pair Programming:** Two developers work together on one machine—one writes code while the other reviews in real-time.
- **Collective Ownership:** No single developer "owns" a module; anyone can modify any part of the codebase.
- **Sustainable development Process:** Avoids burnout by discouraging excessive overtime, ensuring long-term productivity.

AGILE DEVELOPMENT TECHNIQUES

4. Embracing Change

- **Test-First Development (TDD):**
  - Write automated tests **before** coding.
  - Ensures code meets requirements from the start.
- **Refactoring:** Continuously improve code structure **without changing functionality** to maintain simplicity.
- **Continuous Integration (CI):**
  - New code is integrated and tested **multiple times a day**.
  - Catches integration issues early.

5. Simplicity in Design

- **YAGNI ("You Aren’t Gonna Need It"):** Avoid over-engineering by only coding for **current requirements**, not hypothetical future needs.
- **Minimal Documentation**

AGILE PROJECT MANAGEMENT

- An approach to project and team management based on Agile Manifesto.
- The Manifesto is a collection of four values that define the mindset that all Agile teams should strive for.
- provides a structured yet flexible approach to software development that aligns with Agile principles while addressing the need for project visibility and control.
- **Core Principles of Agile Project Management**
- Agile project management emphasize:
  - **Iterative progress** through short development cycles (sprints).
  - **Self-organizing teams** with collective ownership.
  - **Continuous stakeholder collaboration** (e.g., Product Owner involvement).
  - **Adaptability to change** over rigid long-term planning.

AGILE PROJECT MANAGEMENT

The Scrum Framework

- Scrum is the most widely adopted Agile project management method.
- It structures work into fixed-length iterations (sprints) and defines clear roles, artifacts, and ceremonies.

1. Key Roles

Role	Responsibilities
Product Owner	- Represents stakeholders/customers. - Prioritizes the Product Backlog. - Defines acceptance criteria.
Scrum Master	- Ensures Scrum practices are followed. - Removes obstacles for the team. - Facilitates daily stand-ups.
Development Team	- Self-organizing group (5–9 members). - Delivers a potentially shippable increment each sprint.

AGILE PROJECT MANAGEMENT

2. Scrum Terminology

- **Product Backlog** : Prioritized list of features, user stories, and tasks.
- **Sprint Backlog**: Subset of Product Backlog items selected for a sprint. Broken into actionable tasks.
- **Increment**: Working software delivered at the end of each sprint (potentially shippable).
- **Sprint:** A development iteration. Sprints are usually 2 to 4 weeks long.

AGILE PROJECT MANAGEMENT

3. Scrum Events

Event	Purpose
Sprint Planning	Team selects backlog items for the sprint and breaks them into tasks.
Daily Scrum	15-minute stand-up to sync progress ("What I did, what I'll do, blockers").
Sprint Review	Demo of the increment to stakeholders for feedback.
Sprint Retrospective	Team reflects on improvements for the next sprint.

THE SCRUM PROCESS (SPRINT CYCLE)

- **Product Backlog Refinement**
  - The Product Owner prioritizes features, user stories, and technical tasks.
  - Items are refined for clarity (e.g., "As a doctor, I want dose-checking to prevent errors").
- **Sprint Planning**
  - The team selects high-priority items from the backlog.
  - Tasks are estimated (story points/hours) based on **velocity** (past sprint performance).
- **Sprint Execution (2–4 weeks)**
  - Developers work on tasks in a self-organized manner.
  - **Daily Scrums** keep the team aligned.
- **Sprint Review & Retrospective**
  - Stakeholders review the increment and provide feedback.
  - The team identifies process improvements.
- **Repeat**
  - Unfinished work returns to the backlog.
  - The next sprint begins with updated priorities.

AGILE PROJECT MANAGEMENT

- **Benefits of Scrum**

✓ **Transparency** – Progress is visible through sprint reviews and daily stand-ups.

✓ **Flexibility** – Requirements can be reprioritized between sprints.

✓ **Faster Feedback** – Stakeholders see working software frequently.

✓ **Improved Team Morale** – Self-organization fosters ownership and collaboration.

AGILE PROJECT MANAGEMENT

Challenges & Adaptations

1. Distributed Scrum (Remote Teams)

- **Solutions:**
  - **Daily video stand-ups** for alignment.
  - **Shared tools** (Jira, Trello) for backlog tracking.
  - **Continuous Integration (CI)** to sync code changes.

2. Scaling Scrum

  - **Large Projects:** Use frameworks like **SAFe** or **LeSS** to coordinate multiple Scrum teams.
  - **Hybrid Models:** Combine Scrum with Kanban (Scrumban) for continuous flow.

- **3. Role Confusion**
  - **Scrum Master ≠ Project Manager** – Focuses on process, not task assignments.
  - **Product Owner ≠ Business Analyst** – Must have decision-making authority.