Introduction to Software Development Projects

Lecture Notes

School of Computing Communication and Media Studies

Masoud Hamad

CS2113 – Software Development Project
Academic Year 2025

Contents

1 What Makes a Software Development Project?

Software projects emerge when there's an **identified need** and **funding**. Projects vary in duration from months to years and involve **stakeholders**—all people affected by the software, including end users and funding customers.

Key Insight

Modern software development increasingly requires understanding stakeholder needs alongside technical skills. Research indicates most development problems are **non-technical** rather than technical in nature.

1.1 Development Team Structure

Development teams are diverse, self-organized groups (typically under 10 people) that include:

- Product Owners
- UX/UI Designers
- Project Managers
- QA Engineers
- Developers

These teams follow a common **software development process** to work efficiently.

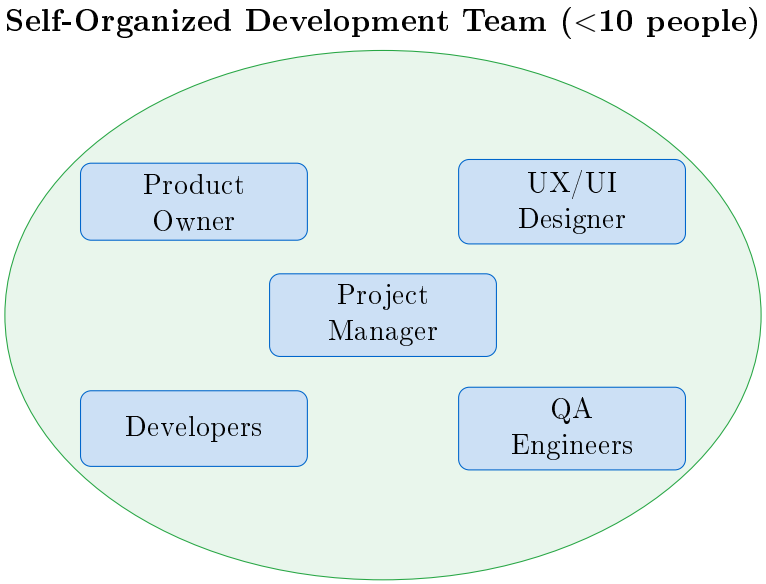

Figure 1: Typical Software Development Team Structure

2 Agile Principles in Software Development

The **Manifesto for Agile Software Development** emphasizes:

Agile Manifesto 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

Key Insight

A critical agile value is **welcoming change**, since requirements frequently shift during development. The Manifesto doesn't specify implementation details; frameworks like **Scrum** and **SAFe** provide concrete processes.

3 Software Development Lifecycle (SDLC)

Six phases structure development:

3.1 The Six Phases

1. **Requirements Phase:** Collecting stakeholder needs into a software requirement specification document
2. **Design Phase:** Analyzing requirements and identifying solutions
3. **Implementation Phase:** Developers code daily tasks toward final product
4. **Test Phase:** Combining automation and manual testing to identify bugs
5. **Deployment Phase:** Distributing software to the production environment (vs. development environment for developers)
6. **Maintenance Phase:** Fixing bugs, addressing issues, monitoring performance

3.2 Waterfall Model

Waterfall Model: A traditional sequential approach completing each phase before the next begins. This approach required all requirements upfront and prevented mid-project changes.

Problem: This approach is problematic because requirements *inevitably shift* during development.

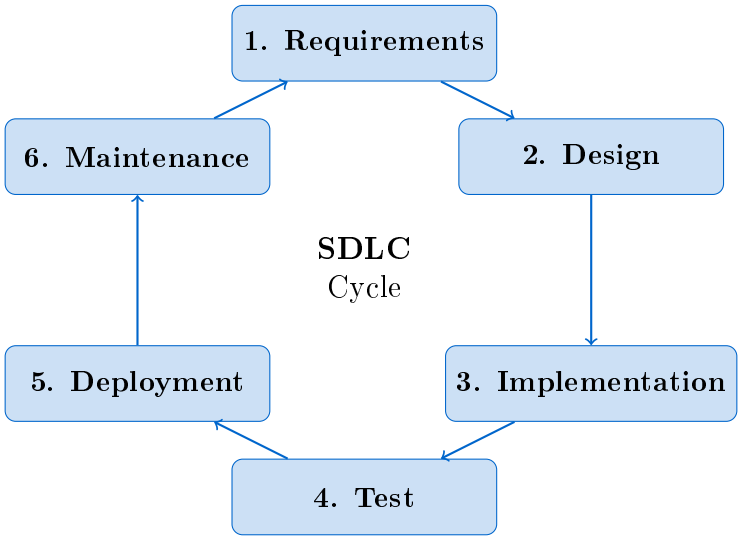

Figure 2: Software Development Lifecycle Phases

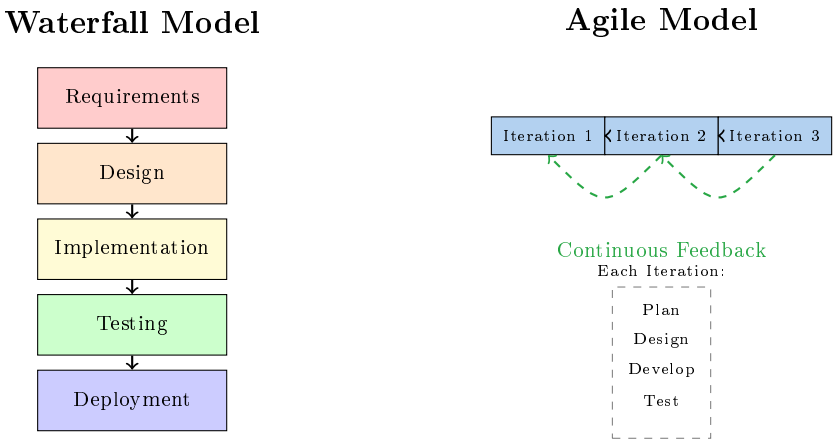

Figure 3: Comparison: Waterfall vs Agile Development Models

4 Agile Software Development Process

Agile uses **iterative approaches** with short iterations (one to two weeks). Each iteration specifies only necessary requirements, allowing flexibility.

Key Insight

Teams implement requirements, producing **working software** users can test and provide feedback on—creating a **feedback loop** essential to agile processes.

Unlike sequential phases, agile completes **all lifecycle phases within each iteration cycle**. This repeating cycle accommodates continuous improvement.

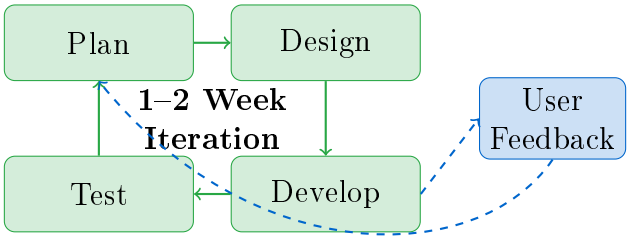

Figure 4: Agile Iteration Cycle with User Feedback Loop

5 Scrum Framework

Scrum is the most widely adopted agile framework (**87%** of respondents in 2022 State of Agile report). The Scrum Guide is the official reference.

The framework defines:

- **Roles** (Scrum Master, Product Owner, Developers)
- **Events** (Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective)
- **Artifacts** (Product Backlog, Sprint Backlog, Increment)

6 Requirement Categories

Requirements divide into two types:

6.1 Functional Requirements

Functional Requirements: User-visible features that describe what the system should do.

Examples:

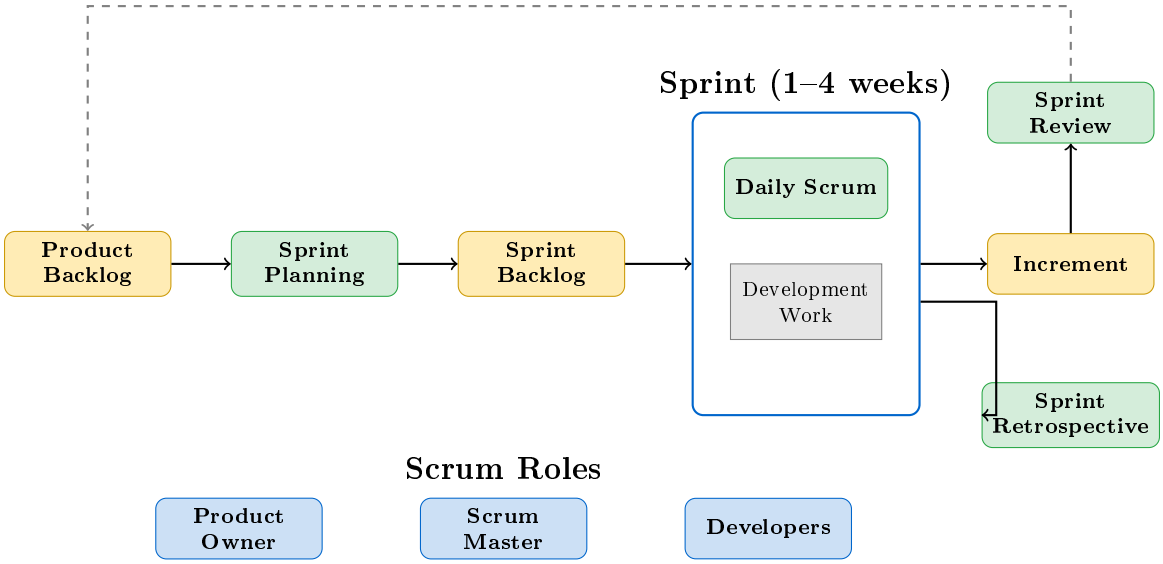

Figure 5: Scrum Framework Overview

- "User should register with username and password"
- "User should see latest blog posts from favorite blogs"

6.2 Non-Functional Requirements

Non-Functional Requirements: Quality constraints invisible to users but addressing security, performance, etc.

Examples:

- "Passwords stored as Bcrypt hashes"
- "Blog post lists load in under one second average"

Requirement Categories

Functional Requirements

- User-visible features
- What system should DO
- Directly testable by users

Non-Functional Requirements

- Quality constraints
- HOW system performs
- Security, Performance, etc.

Figure 6: Functional vs Non-Functional Requirements

7 User Stories

In agile development, functional requirements become **user stories**—short descriptions from the user's perspective. Each sprint, developers implement features based on user

stories.

7.1 Standard Format

User Story Template

"As a **[user persona]**, I want **[action]** so that **[goal]**."

7.2 Examples

Good Example ✓

- "As a **content creator**, I want to **create blogs** so I can **write posts for readers**."
- "As a **blog reader**, I want to **browse blog posts** so I can **find interesting content**."

Key Insight

User stories are brief reminders of features needing implementation, written from users' perspectives **without technical details**.

8 Writing Good User Stories: INVEST Criteria

Strong user stories must:

- Focus on **end-user value**
- Follow the **INVEST criteria**

8.1 INVEST Criteria

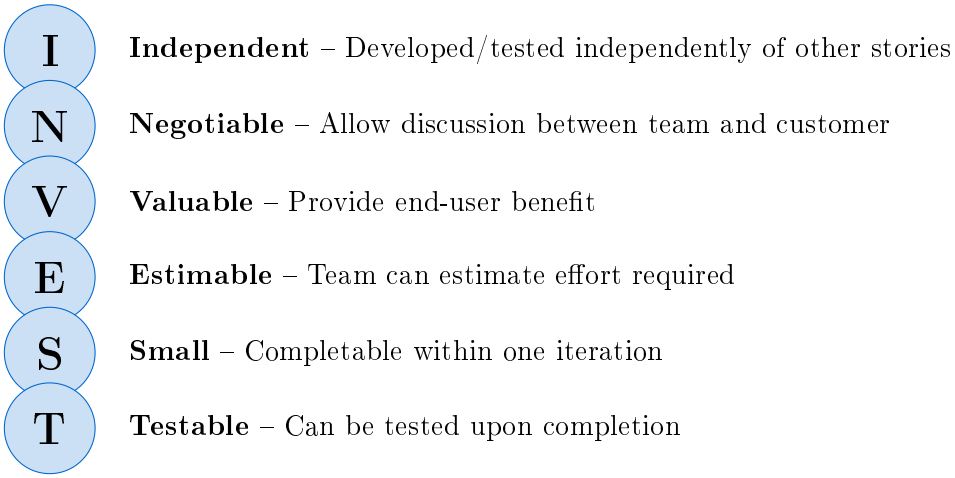

Figure 7: INVEST Criteria for User Stories

8.2 Common Problems and Solutions

8.2.1 Problem 1: Too Large (violates "Small")

Bad Example ×

"As content creator, I want to register with username, password, profile picture, and description"

Good Example ✓

Split into multiple stories:

- Register with username/password
- Add profile picture
- Add profile description

8.2.2 Problem 2: Lacks Technical Clarity (violates "Estimable")

Bad Example ×

"As content creator, I want to export posts from all social media platforms"

Good Example ✓

Narrower scope:
"As content creator, I want to export posts from Medium"

8.2.3 Problem 3: Written from Developer Perspective

Bad Example ×

"As developer, I want to add database index to optimize blog posts table loading"

Good Example ✓

User-focused:
"As blog reader, I want blog post lists loading quickly to find content"

User Story Quality Checklist

- ☐ Written from the user's perspective (not developer's)
- ☐ Focuses on end-user value
- ☐ Small enough to complete in one sprint
- ☐ Can be estimated by the team
- ☐ Independent of other stories
- ☐ Testable with clear acceptance criteria

Figure 8: User Story Quality Checklist

9 Exercises

Submission Requirements

Exercises must be submitted as a **single PDF** to Moodle before **Sunday, January 4, 2026**. Submission is required to confirm course participation.

9.1 Exercise 1: Scrum Team Roles

Identify the roles in a Scrum Team and describe the responsibilities of each role.

9.2 Exercise 2: Sprints and Scrum Events

Explain what a Sprint is and describe the different Scrum events that occur during a Sprint.

9.3 Exercise 3: Product Backlog vs Sprint Backlog

What is the difference between a Product Backlog and a Sprint Backlog? Who is responsible for each?

9.4 Exercises 4–8: Scrum Principles

4. Discuss the three pillars of Scrum (Transparency, Inspection, Adaptation).
5. How do the three pillars support empirical process control?
6. Compare and contrast Scrum with the Waterfall model.
7. What are the advantages of Scrum over Waterfall?

8. In what situations might Waterfall be preferred over Scrum?

9.5 Exercise 9: True or False

Evaluate the following statements about Scrum practices. For each, state whether it is True or False and explain your reasoning.

a) The Scrum Master assigns tasks to team members.

b) The Product Owner can change the Sprint Backlog during the Sprint.

c) Daily Scrum meetings should be limited to 15 minutes.

d) The Sprint Review is the same as the Sprint Retrospective.

e) Only developers attend the Daily Scrum.

9.6 Exercise 10: Critique User Stories

The following user stories have problems. Identify what is wrong with each and explain which INVEST criterion is violated.

a) "As a developer, I want to refactor the codebase so the code is cleaner."

b) "As a user, I want the system to be fast."

c) "As an admin, I want to manage users, create reports, configure settings, and monitor system health."

d) "As a customer, I want to buy products from the website."

9.7 Exercise 11: Improve User Stories

Rewrite the following poorly written user stories to make them better:

a) "The system should have a login feature."

b) "As a developer, I need to add a REST API endpoint."

c) "Users should be able to do everything related to their profile."

9.8 Exercise 12: Create User Stories

Based on a blogging platform project description, create **six user stories** using the proper format:

> "As a [user persona], I want [action] so that [goal]."

Consider different user personas such as:

- Blog reader
- Content creator
- Administrator

9.9 Exercise 13: Prioritize User Stories

Take the six user stories you created in Exercise 12 and prioritize them. Consider:

- Which stories provide the most value to users?
- Which stories have dependencies on other stories?
- Which stories should be implemented first in a Minimum Viable Product (MVP)?

Create a prioritized list and explain your reasoning for the ordering.

9.10 Bonus Exercise: Team Formation

Form team preferences on a collaborative board. Discuss with your classmates:

- What skills do you bring to a team?
- What areas do you want to improve?
- What type of project interests you most?
