Agile Software Project Management

Instructor – Muhammad Sudais

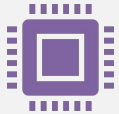
Chapter 3

The Most Popular Agile Methods

Introduction to Agile Methods

Agile methods promise **superior quality products** in **shorter time frames**.

Focus on **satisfying stakeholders** and **end users**.



Two most popular methods: **Scrum** and **Extreme Programming (XP)**.

Other methods covered in Chapter 17.

Scrum Overview

- **Lightweight agile framework** for product development.
- **Empirical process control**: Observing and experimenting.
- **Iterative and incremental development**.
- **Time-boxing**: Fixed time frames for work and meetings.
- **Collaboration**: Stakeholders work together to deliver value.
- **Self-organization**: Teams manage their own work.

Scrum Pillars

Transparency: All project information is visible to stakeholders.

Inspection: Regular checks to ensure progress toward goals.

Adaptation: Adjust processes as needed to address issues.

Scrum Roles

Product Owner: Represents the customer, prioritizes the backlog.

ScrumMaster: Facilitates the Scrum process, removes obstacles.

Scrum Team: Self-organizing, cross-functional team that delivers the product.

Product Owner Responsibilities

- Creates the **product vision**.
- Prioritizes the **product backlog**.
- Accepts or rejects the product increment at the end of each Sprint.
- Ensures the team delivers **maximum business value**.

ScrumMaster Responsibilities

- Protects the team from **external interruptions**.
- Ensures the team follows **Scrum practices**.
- Acts as a **servant leader** to the team.
- Maintains the **Blocks List** (impediments and unresolved issues).

Scrum Team Responsibilities

- **Self-organizing** and **cross-functional**.
- Completes the work during **Sprints**.
- Collaborates with the **Product Owner** to groom the backlog.
- Delivers a **potentially shippable product increment** at the end of each Sprint.

Scrum Artifacts

1. **Product Backlog**: Prioritized list of requirements (user stories).
2. **Sprint Backlog**: Tasks to be completed in the current Sprint.
3. **Sprint Increment**: Potentially shippable product at the end of a Sprint.
4. **Burndown Chart**: Tracks progress during the Sprint.

Product Backlog

- **Dynamic list** of all requirements for the product.
- Prioritized by the **Product Owner** based on **business value**.
- Continuously **groomed** and updated throughout the project.

Sprint Backlog

- Contains the **tasks** selected from the product backlog for the current Sprint.
- Only the **development team** can change the tasks in the Sprint backlog.
- Tasks are **estimated** in terms of complexity, risk, and completion time.

Sprint Increment

- The **potentially shippable product** created at the end of each Sprint.
- Must meet the **Definition of Done** (agreed-upon criteria for completion).
- Presented to the **Product Owner** and stakeholders for acceptance.

Burndown Chart

- Tracks the **remaining work** in the Sprint.
- Shows the **team's progress** toward completing the Sprint backlog.
- Helps predict whether the team will complete the work on time.

Scrum Events

1. **Sprint Planning**: Plan the work for the Sprint.
2. **Daily Scrum**: 15-minute daily meeting to discuss progress.
3. **Sprint Review**: Demo the product increment to stakeholders.
4. **Sprint Retrospective**: Reflect on the Sprint and identify improvements.

Sprint Planning

- **Time-boxed** to 8 hours for a one-month Sprint.
- The team selects **user stories** from the product backlog.
- The team creates a **Sprint goal** and **Sprint backlog**.
- Tasks are **estimated** and assigned.

Daily Scrum

- **15-minute** time-boxed meeting.
- Each team member answers three questions:
  1. What have I completed since the last meeting?
  2. What will I complete today?
  3. Are there any obstacles in my way?

Sprint Review

- Held at the end of each Sprint.
- The team presents the **Sprint increment** to stakeholders.
- Stakeholders provide **feedback** and **accept or reject** the work.
- The **Product Owner** updates the product backlog based on feedback.

Sprint Retrospective

- Held at the end of each Sprint.
- The team reflects on the **process** and identifies **improvements**.
- Focuses on **what went well**, **what didn’t**, and **how to improve**.
- Lessons learned are applied to future Sprints.

Extreme Programming (XP) Overview

- Focuses on **software development best practices**.
- Core values: **Simplicity, Communication, Feedback, Courage, Respect**.
- Iterations last **1-2 weeks**.
- High priority features are developed first.

XP Core Values

1. **Simplicity**: Do only what is necessary.
2. **Communication**: Face-to-face communication is key.
3. **Feedback**: Frequent feedback from stakeholders.
4. **Courage**: Team members are truthful about progress and estimates.
5. **Respect**: Team members respect each other and the customer.

XP Roles

Customer: Defines requirements and sets priorities.

Developer: Writes code and implements features.

Tracker: Monitors progress and team velocity.

Coach: Guides the team on XP practices.

XP Core Practices

1. **Pair Programming**: Two developers work together on the same code.
2. **Test-Driven Development (TDD)**: Write tests before writing code.
3. **Continuous Integration**: Integrate code frequently to detect issues early.
4. **Small Releases**: Deliver working software in small, frequent releases.
5. **Refactoring**: Improve code design without changing functionality.

Pair Programming

- Two developers work together on the same code.
- One writes the code (**driver**), the other reviews it (**navigator**).
- Improves **code quality** and **knowledge sharing**.

Test-Driven Development (TDD)

- Write **tests** before writing the code.
- The initial test will **fail** because the code doesn’t exist yet.
- Write code until the test **passes**.
- Ensures the code meets the requirements.

Continuous Integration

- Developers integrate code into a shared repository **frequently**.
- Automated tests are run to detect **integration issues** early.
- Ensures the codebase is always in a **working state**.

Small Releases

- Deliver **working software** in small, frequent releases.
- Allows for **early feedback** from stakeholders.
- Reduces the risk of delivering a product that doesn’t meet expectations.

Refactoring

- Improve the **design of the code** without changing its functionality.
- Removes **duplicate code**, increases **cohesion**, and reduces **dependency**.
- Ensures the code is **easy to maintain** and **extend**.

Scrum vs. XP Comparison

Scrum	XP
Project management focus	Software development focus
Iterations: 2-4 weeks	Iterations: 1-2 weeks
Self-organizing teams	Pair programming and collective code ownership
Product Owner prioritizes backlog	Customer drives requirements

When to Use Scrum or XP

- **Scrum**: Best for projects requiring flexibility and adaptability.
- **XP**: Best for software development projects with high risk and frequent changes.
- Both methods can be combined for optimal results.

Chapter Summary

- **Scrum** and **XP** are the most popular agile methods.
- **Scrum** focuses on project management, while **XP** focuses on software development.
- Both methods emphasize **iterative development**, **collaboration**, and delivering **value to the customer**.