Sprint 1: Scrum and Project Setup

Lecture 3 Notes

School of Computing Communication and Media Studies

Masoud Hamad

CS2113 – Software Development Project
Academic Year 2025

Contents

1 Sprint Planning

Sprint Planning occurs at the beginning of each sprint with full team participation. The goal is to define what can be delivered in the upcoming Sprint and how that work will be achieved.

1.1 Key Characteristics

- **Duration:** Planning typically takes a few hours maximum
- **Participants:** Entire Scrum Team
- **Output:** Sprint Backlog with selected user stories and tasks

1.2 Roles in Sprint Planning

- **Product Owner:** Prioritizes user stories and clarifies requirements
- **Developers:** Provide feasibility insights and effort estimates
- **Scrum Master:** Facilitates the meeting and ensures process is followed

Key Insight

Requirements tend to change during development, so Sprint Planning focuses on the near future rather than long-term projections. Plan only what you can realistically accomplish in one Sprint.

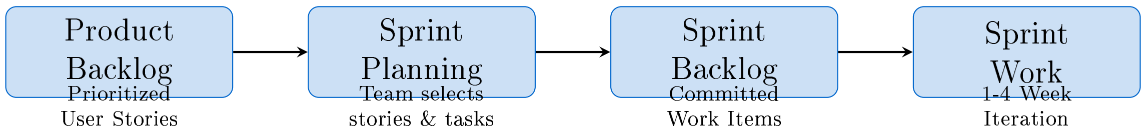

Figure 1: Sprint Planning Flow

2 User Stories and Tasks

2.1 From User Stories to Tasks

User stories are written from the user's perspective **without technical details**. During Sprint Planning, developers split each story into **technical tasks** containing specific implementation requirements.

Tasks are small, technical work items derived from user stories. Each task should be completable in a few hours to a day.

2.2 Example: Breaking Down a User Story

User Story: "As a content creator, I want to create a blog so I can write posts for readers."

Technical Tasks:

1. Create UI form with text fields and submit button
2. Create Blog entity class with JPA annotations
3. Create BlogRepository interface
4. Create BlogController with POST endpoint
5. Configure H2 database connection
6. Write unit tests for BlogService

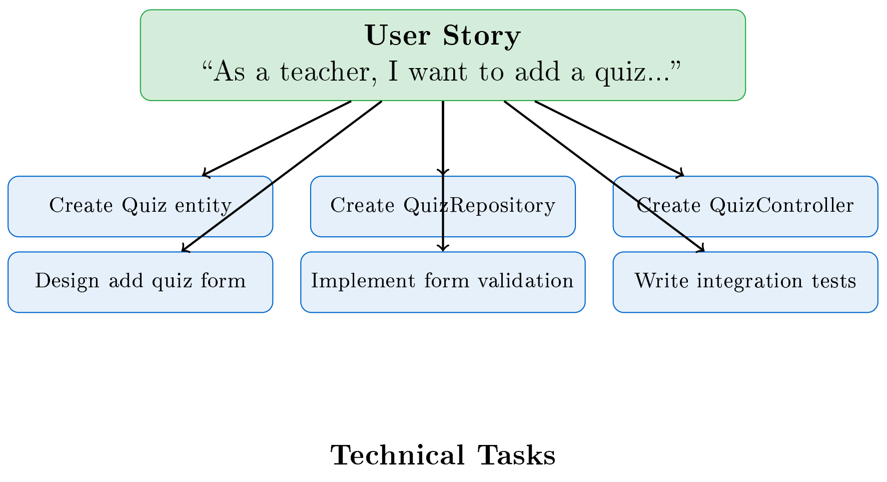

Figure 2: Breaking User Stories into Technical Tasks

3 Scrum Backlogs

3.1 Product Backlog

The **Product Backlog** is a prioritized list of all requirements (primarily user stories) for the product. The Product Owner maintains priorities; highest-priority items appear first.

Characteristics:

- Contains all user stories for the entire project
- Ordered by priority (most important at top)
- Items are removed once implemented and accepted
- Continuously refined and updated

3.2 Sprint Backlog

The **Sprint Backlog** contains the technical tasks selected from the Product Backlog for the current Sprint. It tracks task progress throughout the Sprint.

Sprint Backlog tracks:

- Which user story each task relates to
- Team member assignments
- Task progress (Not Started, In Progress, Done)

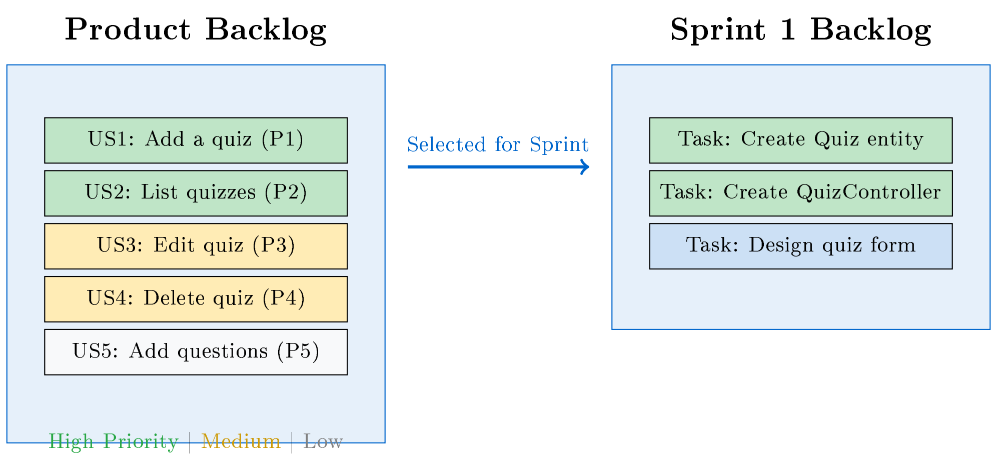

Figure 3: Product Backlog vs Sprint Backlog

4 Task Board and GitHub Projects

4.1 Virtual Task Boards

A task board visualizes work progress using columns representing different states:

Product Backlog	Sprint Backlog	In Progress	Done
Future items	Committed work	Currently working	Completed

4.2 GitHub Projects Setup

1. Navigate to your repository on GitHub
2. Click "Projects" tab
3. Create a new project named "Backlog"
4. Add columns: Product Backlog, Sprint Backlog, In Progress, Done
5. Make the project public

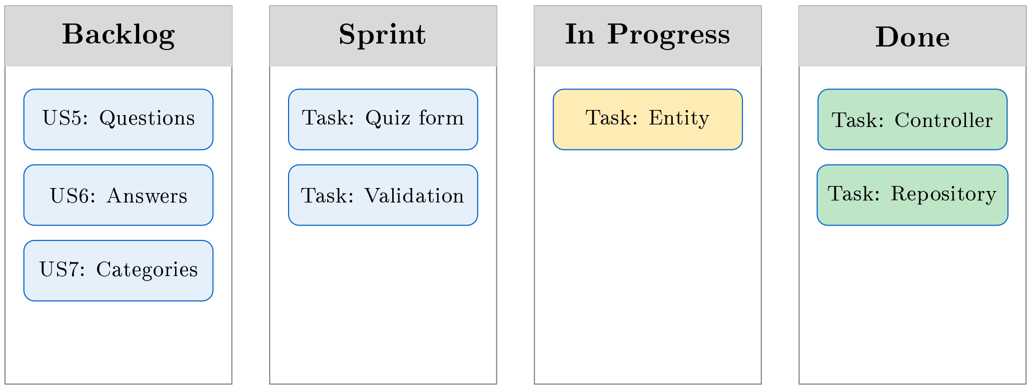

Figure 4: Kanban-style Task Board

5 GitHub Issues and Milestones

5.1 Creating Labels

Labels help categorize issues:

- `user story` – For user story issues
- `bug` – For bug reports
- `frontend` – For frontend tasks
- `backend` – For backend tasks

5.2 Creating Milestones

Milestones group issues for a specific Sprint:

1. Go to Issues → Milestones
2. Create "Sprint 1" milestone
3. Set a due date (end of Sprint)

5.3 Creating Issues

For each user story and task:

1. Set descriptive title
2. Add appropriate label(s)
3. Assign to team member
4. Set milestone (Sprint 1, 2, or 3)
5. Add to Backlog project

6 Daily Scrum

The **Daily Scrum** is a 15-minute daily event where team members synchronize activities and create a plan for the next 24 hours.

6.1 Three Questions

Each team member answers:

1. What did I complete since the last meeting?
2. What will I work on next?
3. Are there any obstacles blocking my progress?

Tip

The Daily Scrum is often called "daily stand-up" because standing discourages lengthy discussions. Keep it short and focused!

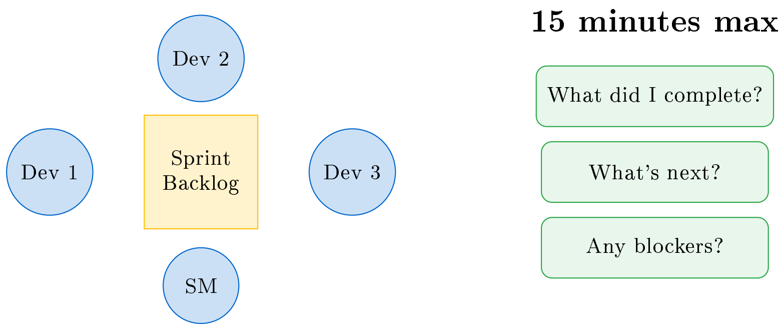

Figure 5: Daily Scrum Meeting Structure

7 Teamwork Best Practices

- Work collaboratively and divide tasks deliberately
- Maintain active communication about progress
- Push code frequently: `git add`, `git commit`, `git push`
- Pull frequently: `git pull`
- Expect and handle merge conflicts
- Adjust Sprint Backlog as needed during implementation

Warning

Tasks aren't immutable! If you discover a task is more complex than expected, discuss with your team and adjust the Sprint Backlog accordingly.

8 REST API Basics

8.1 Introduction to REST Controllers

For the frontend to access backend data, we create REST API endpoints:

```java
@RestController
@RequestMapping("/api")
@CrossOrigin(origins = "*")
public class QuizRestController {

    @Autowired
    private QuizRepository quizRepository;

    @GetMapping("/quizzes")
    public List<Quiz> getPublishedQuizzes() {
        return quizRepository.findByPublishedTrue();
    }
}
```

8.2 CORS (Cross-Origin Resource Sharing)

The frontend (Vite on port 5173) and backend (Spring Boot on port 8080) run on different origins. The `@CrossOrigin` annotation enables cross-origin requests.

8.3 Fetching Data in Frontend

```javascript
fetch("http://localhost:8080/api/quizzes")
    .then(response => response.json())
    .then(quizzes => {
        console.log(quizzes);
    });
```

9 Developer Guide Documentation

The README.md should include clear instructions for starting the application:

```markdown
## Developer Guide

```

```
### Prerequisites
- Java 21 or higher
- Maven

### Running the Backend
1. Clone the repository
2. Navigate to project folder
3. Run: ./mvnw spring-boot:run
4. Access: http://localhost:8080
```

Tip

Test your developer guide by having a team member follow the instructions exactly on a fresh clone of the repository!

10 GitHub Releases

Releases "freeze" source code at specific commit points:

1. Go to repository → Releases
2. Click "Create a new release"
3. Create tag (e.g., `sprint1`)
4. Set title (e.g., "Sprint 1")
5. Describe implemented features
6. Publish release

11 Sprint Review

The **Sprint Review** occurs at the end of each Sprint. The team demonstrates implemented user stories to the Product Owner and stakeholders.

11.1 Sprint Review Requirements

- Working application (ideally deployed)
- Sensible test data (no "test123" placeholders)
- Clear demonstration of features from **user perspective**
- Focus on functionality, not code

12 Exercises

Sprint 1 Deadline

All work must be pushed to GitHub repository before the Sprint deadline.

12.1 Exercise 1: Create Backlog Project

Create a GitHub Project named "Backlog" with columns: Product Backlog, Sprint Backlog, In Progress, Done.

12.2 Exercise 2: Select Scrum Master

Choose a Scrum Master for Sprint 1 from your team.

12.3 Exercise 3: Create Labels

Create a "user story" label in your repository for categorizing user story issues.

12.4 Exercise 4: Create Milestone

Create a "Sprint 1" milestone to group Sprint-related issues.

12.5 Exercise 5: Create User Story Issues

Create GitHub issues for each Sprint 1 user story with appropriate labels and milestone.

12.6 Exercise 6: Database Planning

Plan your database schema:

- Identify required entities
- Define entity attributes
- Determine relationships
- Configure H2 database in `application.properties`

12.7 Exercises 7–10: Task Planning

For each user story (Add Quiz, List Quizzes, Edit Quiz, Delete Quiz):

- Break down into technical tasks
- Create issues for each task
- Assign team members
- Add to Sprint Backlog

12.8 Exercise 11: Assign Team Members

Assign team members to all task issues based on skills and workload balance.

12.9 Exercise 12: Daily Scrum

Conduct Daily Scrum meetings throughout the Sprint, answering the three questions.

12.10 Exercises 13–18: Additional Task Planning

Plan tasks for remaining user stories (Questions, Answer Options management).

12.11 Exercise 19: Initialize Frontend

Create a frontend application (using Vite/React) in your repository.

12.12 Exercise 20: REST Controller

Implement a REST controller method returning only published quizzes.

12.13 Exercise 21: Frontend Quiz List

Implement quiz listing on the student frontend using fetch API.

12.14 Exercise 22: Developer Guide

Write a developer guide in README.md with backend startup instructions.

12.15 Exercise 23: Deploy Backend

Deploy your backend application to the production environment.

12.16 Exercise 24: Create Release

Create a GitHub release named “Sprint 1” with implemented features description.

12.17 Exercise 25: Sprint Review Preparation

Prepare demonstration for Sprint Review:

- Ensure working application
- Prepare test data
- Practice demo from user perspective

Bonus: Feature Branch Workflow

```bash
# Create and switch to feature branch
git checkout -b feature/add-quiz

# Work on feature...
git add .
git commit -m "Implement add quiz feature"

# Push branch to remote
git push --set-upstream origin feature/add-quiz

# Create Pull Request on GitHub
# After review, merge to main
```