Mini-Project: EMLS

Tuomas Vuontisjärvi, Miika Piipurinen

December 9, 2025

We identify two components to case problem. First is summarized as unclean code and the second is the size of the changes. We suggest configuring Source Code management and implementing CI pipelines to fix code quality issues. We also recommend taking into use task management systems and habits.

CI Pipelines

To address the problems, will rely on CI/DevOps principles to provide a solution. First, we would implement an automated CI Pipeline with at least the following tasks:

1. Unit tests with coverage.
2. Static code analysis.
3. Automated builds of binaries and Docker images.
4. Integration tests with the newly built component.

The philosophy behind the Pipeline is to automate as much of the code quality checking and building process as possible. There exists very good Static Code Analysers, such as SonarQube and Coverity, that will flag many of the messy code issues. Unit tests help factor the code into testable and reusable format, while providing test coverage which also help with quality checking. Automating builds and integration tests also ensure that production code changes are committed in a format that supports automated build process.

The pipeline should be configured to block development branch merges when the pipeline status is failed. This ensures that at the very least, the automated checks must pass before changes can be committed. It also promotes making smaller more manageable changes.

The Source Code management should also be configured to block unreviewed merge requests. This adds essentially a human gate to the code. We suggest assigning reviewers with clean coding expertise to make no messy code gets committed to main branches.

Task Management

We rely on agile development as the framework for our solution. Instead of rare and major version updates, agile development creates minimal features and improvements continuously, which is automated by the CI.

We suggest using Jira or some other task management system to address the problem of massive commits. Teams should think about how to split features into smaller more manageable components and create tasks based on that. Then for each merge-request all developers should point to a specific task they are working on, which would also help the reviewer.

The task management should help the modeler to break the changes to smaller chunks, which are also easier to review.

Course Material applicability to task

While the course materials deal with testing, Docker and automated pipelines, they aren't straightforwardly applicable to clean code issues.

Comment on LLM Usage

Initially we wrote around three pages of text for the solution. We then used ChatGPT 5.1 to suggest a summarized version. Based on that summary, we rewrote our text to fit the task length requirement. Tex document formatting help was also used from LLM. No LLM-provided text was used in the assignment.