ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING
SOFTWARE ENGINEERING II
CODE INSPECTION - I TERM 2025

Objectives:

- Use a code quality metric tool for preemptive defect detection.
- Determine Key Characteristics of Code Inspection Tools
- Develop autonomy in choosing, setting up, and applying a code inspection tool.

Requirements

- A codebase in **any programming language** (at least 3 meaningful source files).
- A publicly accessible Git repository (GitHub, GitLab, etc.).
- A compatible static analysis tool (e.g., SonarQube, PMD, ESLint, etc.).

Introduction

Code inspection is a systematic and static review of source code to detect quality, design, security, and maintainability problems without executing the program.

Unlike simple syntax checks or linters, code inspection tools identify deeper issues, such as:

- Poor structure or high complexity
- Code duplication
- Performance bottlenecks
- Security vulnerabilities
- Violations of design principles (e.g., coupling, cohesion)
- Maintainability concerns

Code inspection is a core practice in preventive quality assurance, allowing teams to reduce technical debt early and enforce clean, consistent, and secure codebases.

Tools for Code Inspection

You are required to find and configure a code inspection tool that supports your project’s language.The tool must:

- Perform static analysis
- Support report generation
- Categorize issues by type or severity
- Provide actionable feedback (e.g., remediation guidance, rule explanations)

Examples of Tools:

Language	Code Inspection Tools
Java	PMD, SpotBugs, SonarQube
Python	SonarQube, CodeClimate, Prospector
JavaScript	SonarQube, CodeClimate
C#	SonarQube, ReSharper InspectCode
C/C++	SonarQube, CppDepend, Clang-Tidy
Ruby	SonarQube, CodeClimate

Your responsibility is to research, install, and use the appropriate inspection tool for your chosen language.

PMD is a **source code analyzer**. It finds **common programming flaws** like unused variables, empty catch blocks, unnecessary object creation, and so forth [1]. Like other tools, PMD can **verify that coding conventions and standards are followed.** PMD is more focused on **preventive defect detection**. It comes with a vast set of rules and is highly configurable. PMD can also configure - in a simple way - particular rules to use in a specific project [2].

PMD integrates well with IDEs such as Eclipse and NetBeans, and it also fits well into the build process thanks to its smooth integration with Ant and Maven [2].

SonarQube

SonarQube supports **multiple languages** and provides advanced insights like:

- **Code smells**
- **Bugs**
- **Security vulnerabilities**
- **Duplicated code**
- **Technical debt estimation**

It integrates with CI/CD pipelines and supports rulesets for dozens of languages. SonarQube presents its reports via a web interface and emphasizes **team-based quality tracking**.

Activity

You will:

1. Find and set up an appropriate static code analysis tool for your project’s language.
2. Perform an inspection of your own or a peer’s repository.
3. Fix defects and report your results in an iterative process.

1. Use or Create a Codebase

- Use a project you’ve already written (for this course or another) or collaborate with a peer.
- Clone the project to your local environment.
- Ensure it is under version control and publicly accessible.

2. Research and Set Up a Static Analysis Tool

- **You must research and configure a static code analysis tool** that supports your project’s language.
- Your tool **must generate reports** and highlight at least:
    - Code smells or maintainability issues
    - Bugs or defects
    - Security vulnerabilities (if any)

You **cannot skip this** step. Tools such as PMD, SonarQube, ESLint, etc., must be integrated and used.

Hints:

- Start at https://rules.sonarsource.com/ to explore supported languages and tools.
- Read only official documentation for your chosen tool (e.g., PMD’s or SonarQube’s site).

3. Analyze the Code

- Perform a full scan with the static tool.
- Take a screenshot of the **initial analysis report** (with visible violations).
- Document the most important findings (name of rule, file, line, and a short description).

4. Fix Detected Issues

- Fix **at least five significant issues**, including:
    - One **code smell**
    - One **bug**

    - One **security hotspot or vulnerability** (if any present)

- Add meaningful commit messages for each fix.

  Avoid just disabling rules or suppressing warnings unless absolutely necessary — if you do, justify them in your report.

Deliverables

Lab report including:

- Initial and final report screenshots
- URL to the Git repository used
- Static tool setup summary (tool name, how it was configured, what language it supports)
- Summary of peer review process (who reviewed what, suggestions made)
- Reflection (What did you learn about your own code? What was surprising?)

Both **tool-generated reports** (before and after)

Tool configuration file(s), e.g., sonar-project.properties, .eslintrc.json, pmd.xml, etc.

Rubric

Criteria	Points
Lab Report Quality	40
- Clear screenshots, explanations, and project URL	
Issue Fixes (Code Improvements)	20
- At least 5 meaningful violations addressed	
Tool Setup and Use	20
- Correct setup of a static analysis tool (not skipped or faked)	
Challenge Compliance	20
- Own repo used, no base repo, tool chosen independently	
Penalty: No public repo URL	-100

Late Submission Policy

Delay (§)	Penalty (Ω)
1 hour or less	loss of 10%

1 to 6 hours	loss of 20%
6 to 24 hours	loss of 30%
Over 24 hours:	loss of 100%

(§) every clock hour counts including weekends or holidays.
(Ω) automatic and non-negotiable penalty

References

[1] PMD, "PMD Source Code Analyzer," [Online]. Available: https://pmd.github.io/.

[2] J. Ferguson, Java Power Tools, O'Reilly Media, 2008.

[3] PMD, "Java Rules," [Online]. Available: https://pmd.github.io/latest/pmd_rules_java.html.

[4] PMD, "Suppressing warnings," [Online]. Available: https://pmd.github.io/latest/pmd_userdocs_suppressing_warnings.html.

[5] Instructions, https://github.com/leortyz/softwareEngineeringResources/wiki/Code-Inspection