

COMPREHENSIVE TECHNICAL DOCUMENTATION: DELIVERY 2

Governance Pipeline & Tech Debt Audit

Project: Pdds-ISA-Project (Spring PetClinic)

Engineer: Pablo Pineda & Christian Martinez

Submission Date: February 16nd, 2026

1. Governance Infrastructure (CI Workflow)

An automated quality control system was implemented using GitHub Actions to ensure software integrity. This system acts as a 'Quality Gate' that evaluates every code change before final integration.

1.1 Tools and Technologies:

- **GitHub Actions:** CI/CD orchestrator for executing governance rules.
- **SonarCloud:** Static code analysis and metrics centralization.
- **Checkstyle:** Coding standards auditing (Sun Checks).
- **JaCoCo:** Unit test coverage measurement.
- **Maven:** Dependency management and build automation.

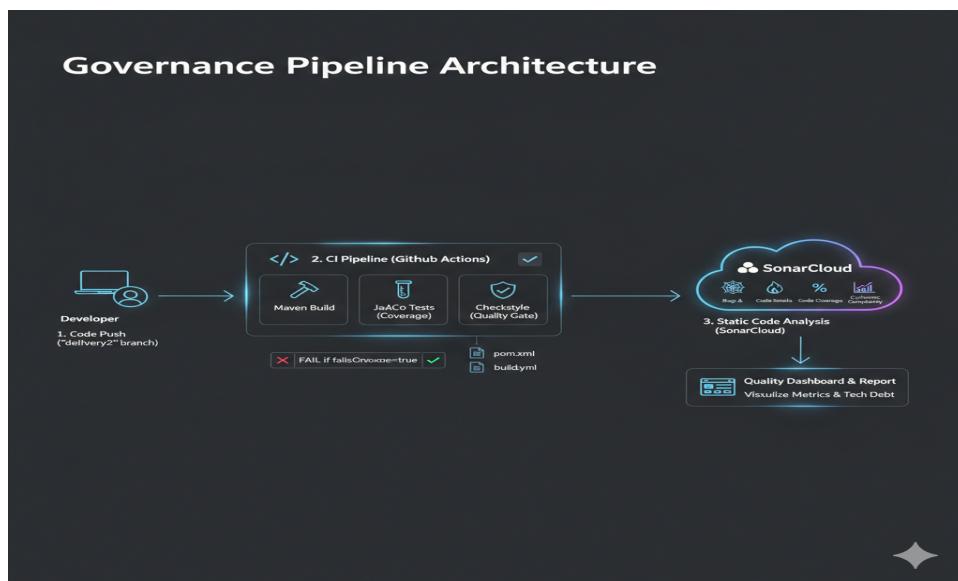


Figure 1.1: Governance Pipeline Architecture Diagram.

1.2 Specific Configurations:

- **build.yml:** Redundant workflows were removed, and the analysis command was configured to force visibility on the SonarCloud main branch (-Dsonar.branch.name=main).
- **pom.xml:** Plugin integration with 'failsOn Error=false' to allow a comprehensive debt report without interrupting the dashboard visibility flow.

2. Tech Debt Audit (Hotspots Identification)

Using pipeline data, an audit was conducted to identify 'Hotspots': files where high complexity and change frequency (churn) generate operational risks.

2.1 Identified Hotspots (Cyclomatic Complexity > 20):

- **OwnerController.java:** Identified as a critical point due to excessive search and edit logic (Fat Controller).
- **PetController.java:** High structural fragility caused by tight coupling in validation management.
- **Owner.java:** Entity with structural complexity exceeding recommended standards for data models.

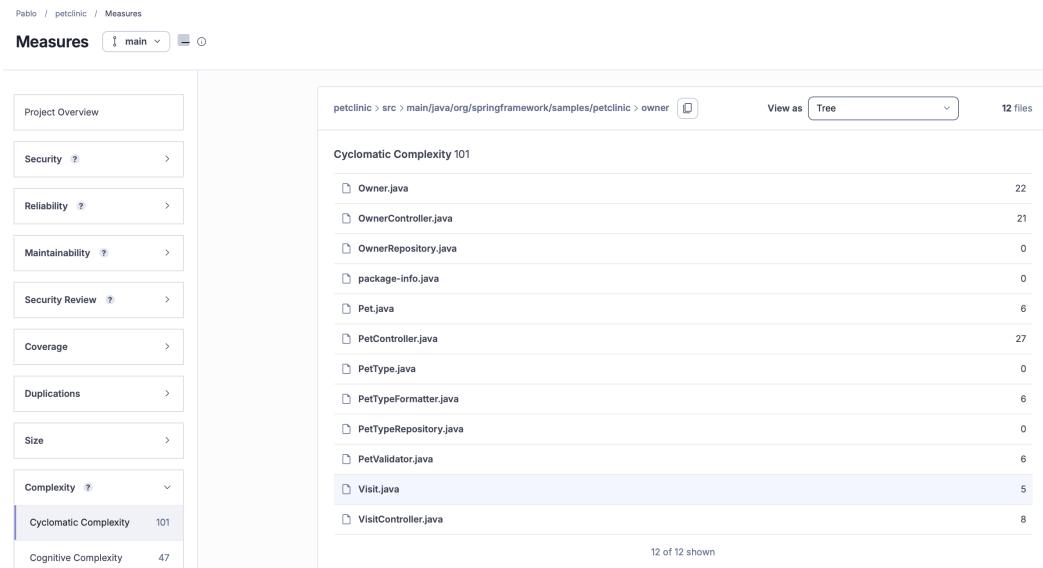


Figure 2.1: Complexity analysis and metrics in SonarCloud (mesure.png).

3. Refactoring Plan: Strangler Fig Pattern

To mitigate the identified debt, a controlled migration towards a cleaner architecture is proposed:

- **Phase 1 (Interception):** Creation of a decoupled Service Layer to extract logic from the controllers.
- **Phase 2 (Strangling):** Gradual migration of complex methods, validated at each step by the CI Pipeline and JaCoCo.
- **Phase 3 (Elimination):** Final removal of legacy code once SonarCloud metrics validate the new structure.

4. Execution Evidence

Figure 4.1: Successful Pipeline State in GitHub Actions (verde.png)

The screenshot shows the GitHub Actions interface for the PabloPI50 / Pdds-ISA-Project repository. The 'Actions' tab is selected, displaying a list of workflow runs under the heading 'All workflows'. The left sidebar contains sections for Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area shows seven workflow runs for the 'SonarQube' pipeline:

- fix: route delivery2 analysis to main dashboard**: Status: delivery2, Event: 29 minutes ago, Duration: 2m 3s.
- feat: final sonar command with organization key and clean workflow**: Status: delivery2, Event: Today at 2:36 PM, Duration: 2m 32s.
- fix: force disable all checkstyle failures to allow sonar sync**: Status: delivery2, Event: Today at 2:25 PM, Duration: 44s.
- fix: force disable all checkstyle failures to allow sonar sync**: Status: delivery2, Event: Today at 2:25 PM, Duration: 32s.
- change to permit failsOnError**: Status: delivery2, Event: Today at 2:10 PM, Duration: 58s.
- feat: complete SonarQube integration with pom.xml and build wor...**: Status: delivery2, Event: Today at 1:57 PM, Duration: 49s.
- feat: Add Governance Workflow**: Status: delivery2, Event: Today at 1:01 PM, Duration: 39s.

Figure 4.2: Consolidated Quality Dashboard in SonarCloud (sonar.png)

The screenshot shows the SonarCloud dashboard for the petclinic project. The left sidebar includes sections for Overview, Analysis, Issues (selected), Security Hotspots, Reporting, Measures, Activity, Policies, Quality Profiles, Quality Gate, Project, Pull Requests, Branches, Code, Project Information, and Administration. The main area displays the 'Issues' section with a summary of findings:

- Software quality**:
 - Security: 0
 - Reliability: 0
 - Maintainability: 21
- Severity**:
 - Blocker: 3
 - High: 3
 - Medium: 9
 - Low: 2
 - Info: 4
- Code attribute**:
 - Consistency: 3
 - Intentionality: 14
 - Adaptability: 4
 - Responsibility: 0
- Type**:
 - Bug: 0
 - Vulnerability: 0
 - Code Smell: 21
- Type Severity**:
 - Blocker: 3
 - Critical: 3
 - Major: 9
 - Minor: 1

A prominent yellow warning message at the top states: "Last analysis had a warning". Below it, a list of specific code smells found in various Java files (src.../samples/petclinic/module/NamedEntity.java, src.../samples/petclinic/owner/Owner.java, src.../samples/petclinic/owner/OwnerController.java, src.../samples/petclinic/owner/PetController.java, src.../samples/petclinic/system/CrashController.java) is shown, each with its severity, effort, and a 'View warning' link.