

Continuous Code Inspection with SonarQube

Prof. Luigi Libero Lucio Starace

luigiliberolucio.starace@unina.it

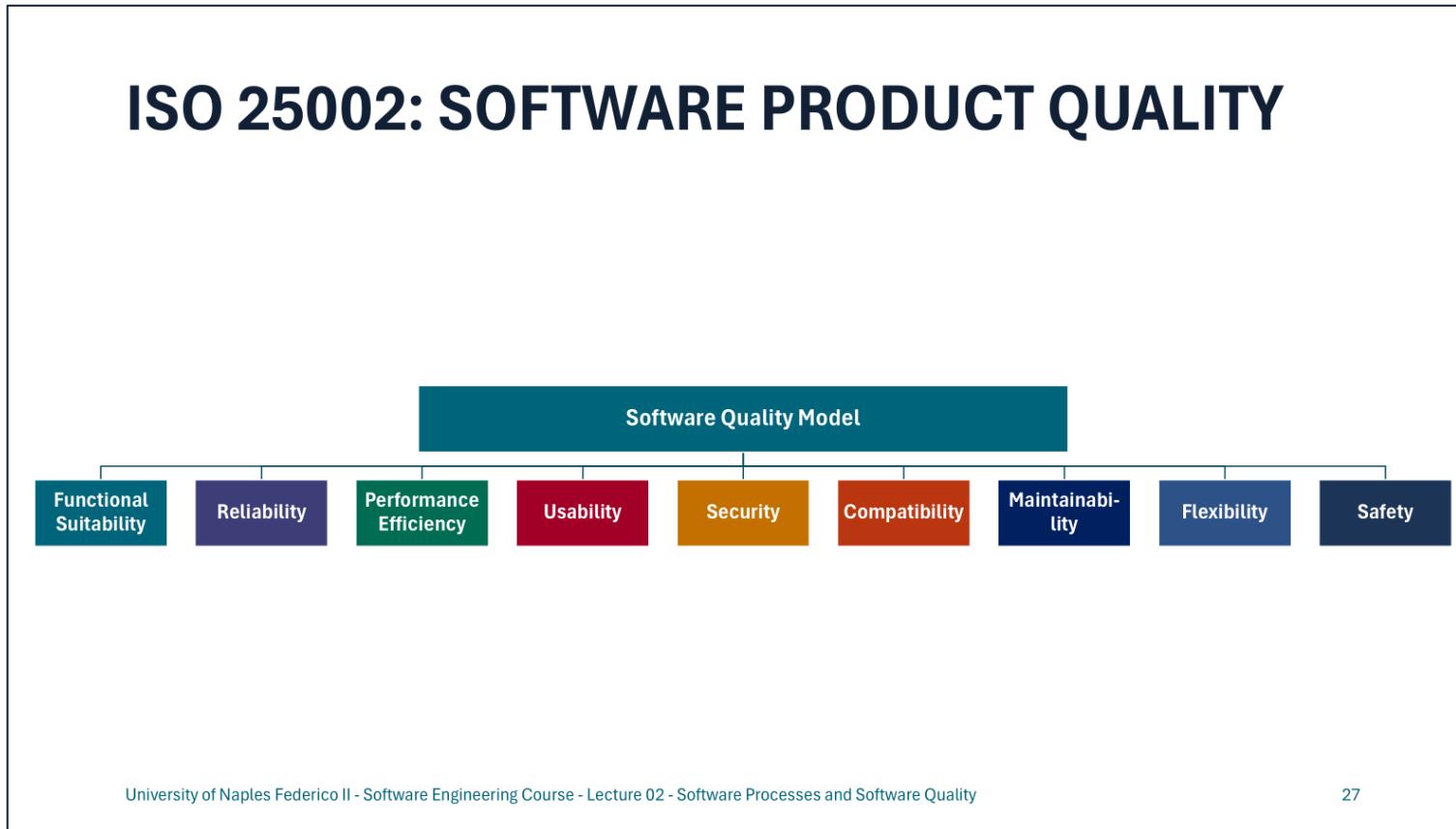
<https://luistar.github.io>

<https://www.docenti.unina.it/luigiliberolucio.starace>



(A look back on) Software Quality

- Back in Lecture 2, we discussed about **Software Quality**

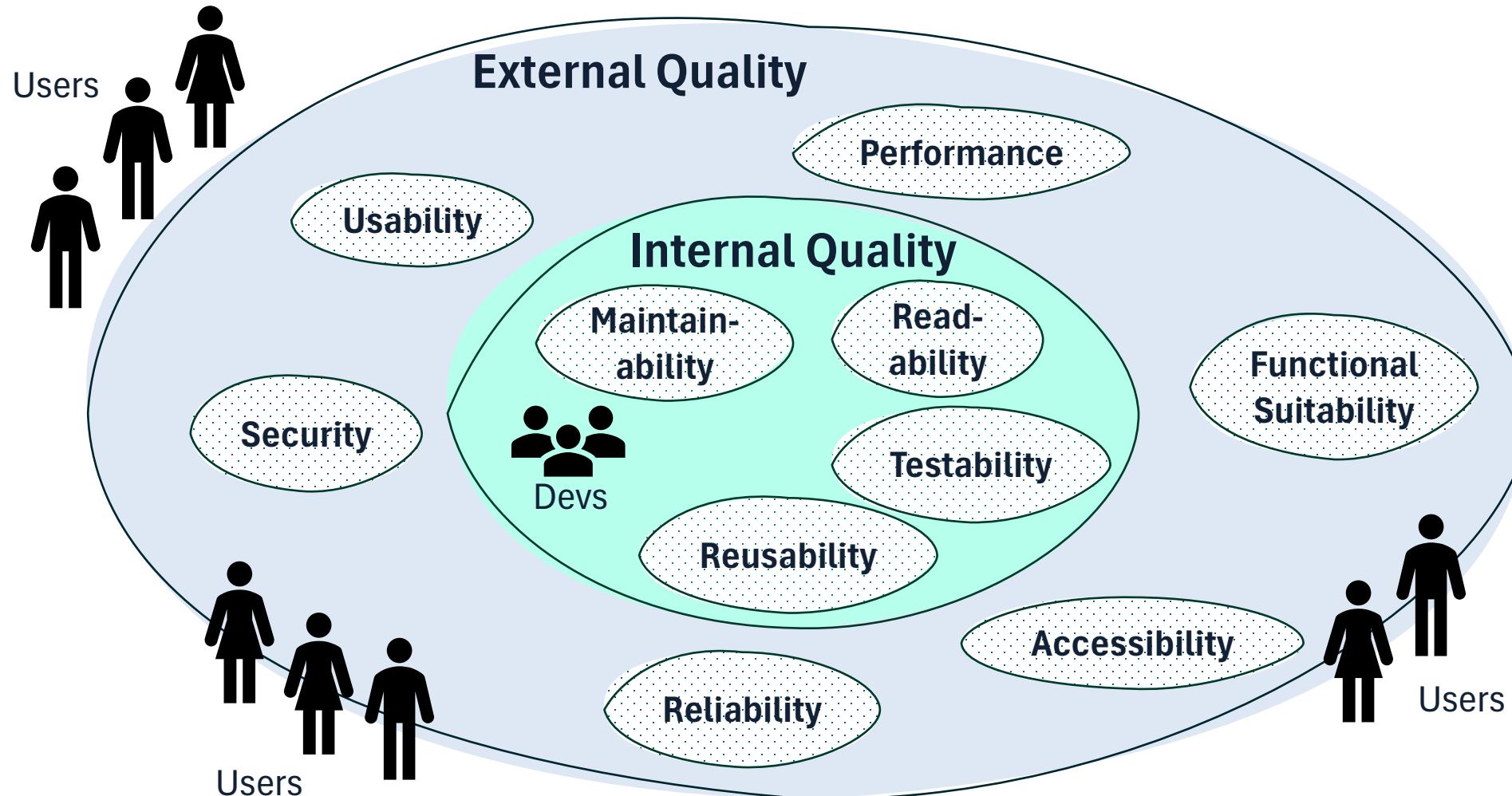


Internal and External Software Quality

Quality characteristics can also be conceptually organized in:

- **Internal Software Quality** characteristics
 - Quality as perceived by developers
 - A software with high internal quality has a codebase which is easy to read, understand and evolve
- **External Software Quality** characteristics
 - Quality as perceived by users
 - E.g.: usability, functional suitability, efficiency

Internal and External Software Quality



Internal and External Software Quality

- External quality is perceived **directly** by users and customers
- Internal quality is **not directly** perceived by users and customers
 - But it has a direct impact on the costs required to evolve a software system
 - It also has a direct impact on some external qualities. For example, **testability** has an impact on guaranteeing **functional correctness**



House with a beautiful exterior (external q.),
but shaky foundation (internal q.)

Origins of poor internal quality

- **Pressure to meet fast-paced deadlines**
- **Inadequate knowledge:** Developers need more experience or training.
- **Manual issue remediation:** Without automated tools, identifying and fixing issues can be inconsistent and error-prone.
- **Inconsistent coding styles:** Varied coding practices within a team can result in a codebase that's difficult to update efficiently.
- **AI coding assistants:** While promising efficiency, these tools can introduce buggy and insecure code if not properly managed

Impact of poor internal quality

- **Reduced maintainability and scalability:** Bad code is hard to understand and modify, making it difficult to evolve.
- **Increased bug count and technical debt:** Poorly written code is prone to bugs, contributing to technical debt that accumulates over time.
- **Decreased productivity and efficiency:** Developers spend more and more time deciphering and fixing bad code, diverting focus from innovation and new functionality.
- **Increased costs and risks:** The cumulative impact of bad code results in higher maintenance costs, frequent bug fixes, rework, and increased technical debt. Additionally, it poses risks to software reliability, security, and stability, leading to reputational damage and compliance issues.

Costs of Poor Internal Software Quality

- The Consortium for IT Software Quality reported that, in 2022 [1]:
 - Unsuccessful development projects costed up to **\$260 billion**
 - The cost of poor software quality in the U.S. grew to at least **\$2.41 trillion**
 - The accumulated technical debt has grown to **\$1.52 trillion**
- According to the Standish Group's CHAOS Report [2], **only 31% of software projects are completed on time and within budget**, with bad code being a significant factor.

[1] <https://www.it-cisq.org/wp-content/uploads/sites/6/2022/11/CPSQ-Report-Nov-22-2.pdf>

[2] <https://www.csus.edu/indiv/v/velianitis/161/chaosreport.pdf>

Measuring Software Quality

- Remember:
 - *You can't manage what you can't measure*
 - *You can't improve what you can't measure*
- We already discussed ways to measure external quality attributes
- **How do we manage and deal with internal quality?**

Code Inspections

- One way would be to perform **code inspections**
- A dedicated team goes over the source code, reads it, and ensures it adheres to principles detailed in a given **checklist**
- For example, for each Java source file
 - ❑ There is no duplicated code
 - ❑ All method names use the camel case naming convention
 - ❑ No method has more than 5 arguments
 - ❑ All variable names use the camel case naming convention
 - ❑ All opened resources are properly closed
 - ❑ No method is longer than 50 LoCs
 - ❑ ...

Code Inspections

Performing manual code inspection does not scale well

- What if our codebase is **large** (e.g.: **100k+ LoCs**)?
- What if changes are made multiple (e.g.: **1k+**) times a day?
- Performing inspections manually is **unfeasible!**
 - Luckily, Software Engineers developed tools to assist with that
 - You'll see that there's way more to development than a good old IDE!

Automated Code Inspections Solutions

- Today, we'll take a look at a family of widely-used automated code inspection solutions by SonarSource



Issues and Rules

- Code is automatically processed to detect **issues**
- **Issues** are basically violations of rules
- The set of rules to apply is customizable
 - The default rules for Java code are [are available here](#)
- There are three types of issues:
 -  **Bug:** Mistakes that can lead to errors or unexpected behavior at runtime.
 -  **Vulnerability:** A point in your code that's open to attack.
 -  **Code Smell:** A maintainability issue that makes your code confusing and/or difficult to maintain.

Issue Severity

- **BLOCKER:** Bug with a high probability to impact the application in production. E.g.: memory leaks, or unclosed JDBC connections.
- **CRITICAL:** Either bugs with low probability to impact the application in production or an issue that represents a security flaw. E.g.: empty catch blocks or SQL injection.
- **MAJOR:** Quality flaws that can highly impact developer productivity. E.g.: Untested code, duplicated code, or unused parameters.
- **MINOR:** Quality flaws that can slightly impact developer productivity. E.g.: lines that are too long, "switch" statements with less than 3 cases.
- **INFO:** Neither a bug nor a quality flaw, just a finding.

A few words before we start

- You now know what a code smell is
- How many code smells would you say there were in the average object orientation project?



0 response submitted

Quanti code smell sono presenti nel progetto medio di Object Orientation?

Scan the QR or use
link to join



[https://forms.office.com/
e/hFz9mX8NZB](https://forms.office.com/e/hFz9mX8NZB)

Copy link



Treemap

Bar



1 of 1



Wandering how do I know that?



Automatic Assessment of Architectural Anti-patterns and Code Smells in Student Software Projects

Marco De Luca
marco.deluca2@unina.it
University of Naples Federico II
Naples, Italy

Sergio Di Meglio
sergio.dimeglio@unina.it
University of Naples Federico II
Naples, Italy

Anna Rita Fasolino
fasolino@unina.it
University of Naples Federico II
Naples, Italy

Luigi Libero Lucio Starace
luigiliberolucio.starace@unina.it
University of Naples Federico II
Naples, Italy

Porfirio Tramontana
ptramont@unina.it
University of Naples Federico II
Naples, Italy

ABSTRACT
When teaching Programming and Software Engineering in Bachelor's Degree programs, the emphasis on creating functional software projects often overshadows the focus on software quality, a trend consistent with ACM curricula recommendations. Dedicated Software Engineering courses take typically place in the later stages of the curriculum, and allocate only limited time to software quality, leaving educators with the difficult task of deciding which quality aspects to prioritize. To educate students on the importance of developing high-quality code, it is important to introduce these skills as part of the assessment criteria. To this end, we have implemented a pipeline based on advanced frameworks such as ArchUnit and SonarQube. It was successfully tested on a class of students engaged in the Object Oriented Programming course, demonstrating its usefulness as a resource for educators and providing some concrete evidence of quality problems in student projects.

28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3661167.3661290>

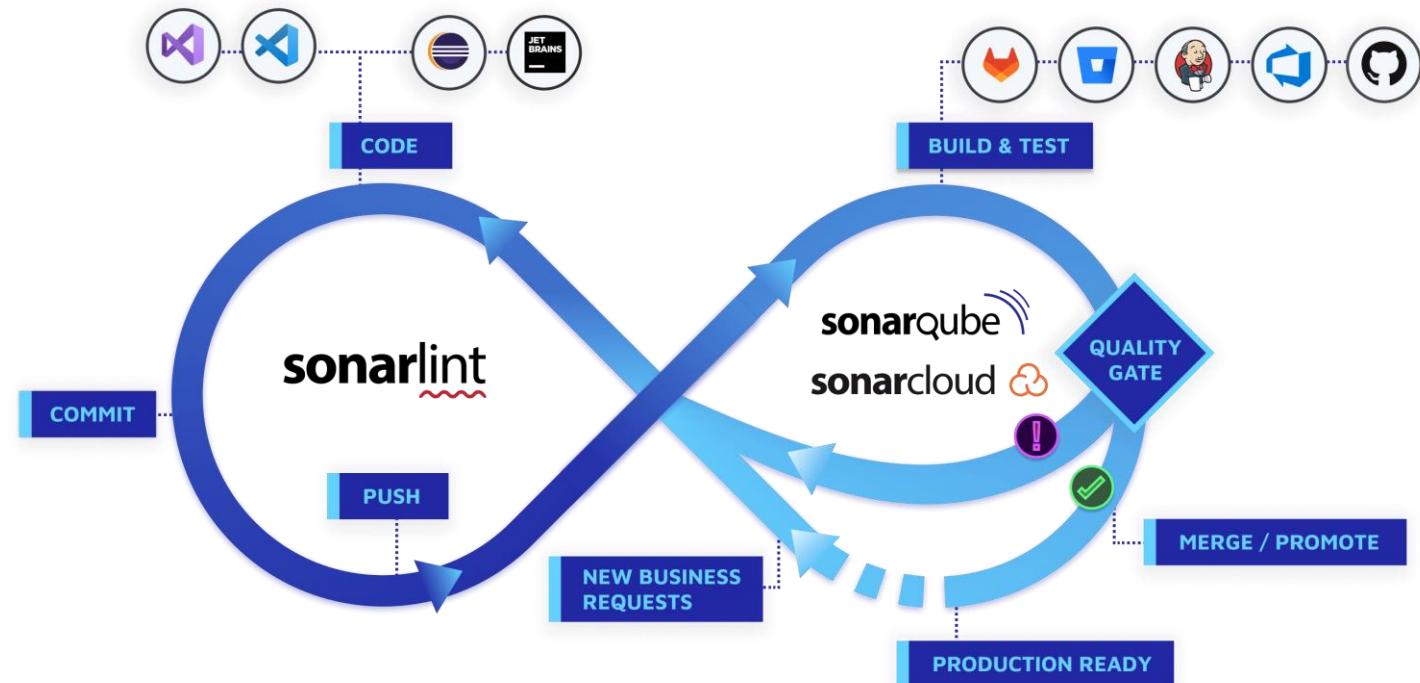
1 INTRODUCTION
The teaching of Programming and Software Engineering in Bachelor's Degree programs, such as Computer Science and Computer Engineering, often emphasizes the ability to create functional projects rather than focusing on software quality. This approach is consistent with the ACM Computer Science and Computer Engineering curricula recommendations for Bachelor degrees [2]. According to these recommendations, software quality is only marginally addressed in typical three-year Bachelor's degree programs, with introductory CS1 courses focusing mainly on programming aspects, and some preliminary software quality concepts being introduced only later in the program, in Software Engineering courses.

<https://dl.acm.org/doi/pdf/10.1145/3661167.3661290>

Automated Code Inspection approaches

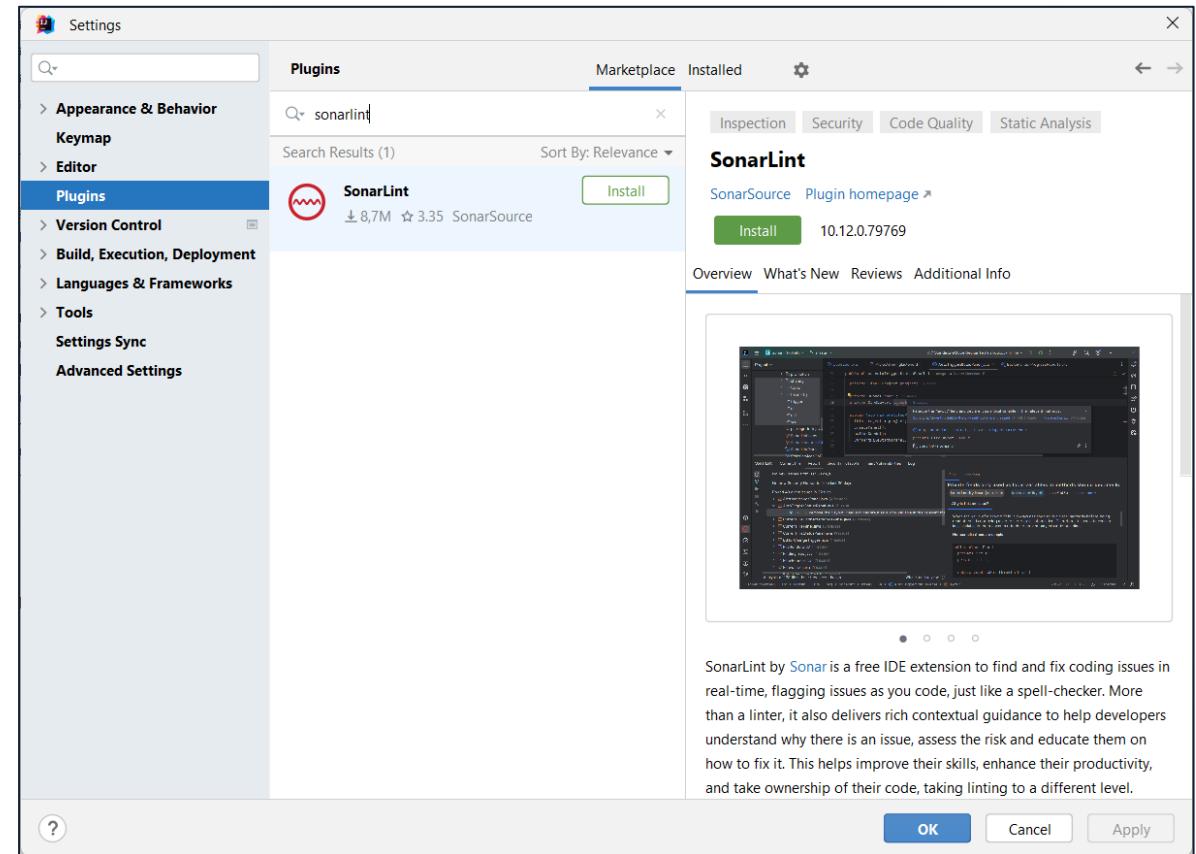
Two (not mutually exclusive) approaches are possible:

- We run the automated code inspections in our IDE, **while** we write code
- Automated code inspections are executed **after** we write code



SonarQube IDE (previously SonarLint)

- Clean as you code approach
- Detect issues as soon as we are writing the code
- Free IDE plugin available for:
 - Jetbrains IDEs
 - Visual Studio
 - Visual Studio Code
 - Eclipse



Installing SonarQube IDE in IntelliJ IDEA

The screenshot shows the IntelliJ IDEA interface with the SonarQube plugin integrated. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and unina-delivery - AggiungiImporto.java. The left sidebar displays the Project structure with the 'target' folder highlighted in yellow. The main editor window shows the Java code for 'AggiungiImporto.java'. Below the editor, the SonarLint tab is active, showing 982 issues found in 51 files, with 27 issues in 'AggiungiCartaFrame.java'. A detailed list of these issues is provided. To the right, the 'Rule Locations' panel is open, specifically for the rule 'String literals should not be duplicated'. It shows the rule details, adaptability (Maintainability), and a note about why it's an issue. The 'Parameters' section includes a threshold of 3. The bottom status bar indicates a Microsoft Defender configuration warning and system information like time and encoding.

unina-delivery > src > AggiungiImporto

Project unina-delivery [O-OProgettoUninaDelivery-Object]

AggiungiImporto.java

```
public class AggiungiImporto extends JFrame {    public AggiungiImporto(GestoreApplicazione ga) {        txtGetSaldo.addKeyListener(new KeyAdapter() {            public void keyTyped(KeyEvent e) {                setVisible(false);                txtGetSaldo.setText("");                JOptionPane.showMessageDialog( parentComponent: null, message: "Importo aggiunto con successo", title: "", JOptionPane.INFORMATION_MESSAGE );            }        });    }    public void actionPerformed(ActionEvent e) {        if (e.getSource() == btSaldo) {            txtSaldo.setEditable(true);        }    }}
```

SonarLint: Current File Report Security Hotspots Taint Vulnerabilities Log

Found 982 issues in 51 files

AggiungiCartaFrame.java (27 issues)

- (29, 15) Remove this unused "bluLink" private field. 4 minutes ago
- (16, 7) Remove this unused import 'javax.swing.JCheckBox'. 4 minutes ago
- (3, 7) Remove this unused import 'java.awt.EventQueue'. 4 minutes ago
- (251, 19) Remove the unnecessary boolean literal. 4 minutes ago
- (244, 243) Remove the unnecessary boolean literal. 4 minutes ago
- (239, 100) Remove the unnecessary boolean literal. 4 minutes ago
- (234, 102) Remove the unnecessary boolean literal. 4 minutes ago
- (240, 3) This branch's code block is the same as the block for the branch on line 235. [+1 location] 4 minutes ago
- (227, 19) Remove the unnecessary boolean literal. 4 minutes ago
- (218, 19) Remove the unnecessary boolean literal. 4 minutes ago
- (209, 19) Remove the unnecessary boolean literal. 4 minutes ago
- (211, 48) Remove the unnecessary boolean literal. 4 minutes ago
- (201, 17) Refactor this method to reduce its Cognitive Complexity from 24 to the 15 allowed. [+20 locations] 4 minutes ago
- (155, 40) Remove the unnecessary boolean literal. 4 minutes ago
- (50, 34) Use static access with "javax.swing.WindowConstants" for "HIDE_ON_CLOSE". 4 minutes ago
- (31, 29) Make "myGestore" transient or serializable. 4 minutes ago
- (28, 16) Rename this field "Mano" to match the regular expression "^[a-z][a-zA-Z0-9]*\$". 4 minutes ago
- (67, 36) Define a constant instead of duplicating this literal "PT Sans" 13 times. [+13 locations] 4 minutes ago
- (205, 59) Define a constant instead of duplicating this literal "ATTENZIONE" 8 times. [+8 locations] 4 minutes ago
- (0, 0) Move this file to a named package. 4 minutes ago
- (211, 3) Merge this if statement with the enclosing one. [+1 location] 4 minutes ago
- (220, 3) Merge this if statement with the enclosing one. [+1 location] 4 minutes ago

Analysis of all project files done 3 minutes ago

What's in this view ?

Current File

Project

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help unina-delivery - AggiungiImporto.java

13 Indexing...

Maven

Endpoints

AI Assistant

Database

Notifications

Coverage

Microsoft Defender configuration: The IDE has detected Microsoft Defender with Real-Time Protection enabled. It might severely degrade IDE performance. It is recommended to add the following paths to the Defender folder exclusion... (5 minu

57:21 LF UTF-8 Tab*

SonarQube Server

- The easiest way to set up a local instance of **SonarQube Server** is to use Docker and the [official SonarQube images](#)

```
@luigi → D/0/T/S/sonar $ docker run -d --name sonarqube  
-e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:10.7-community
```

- Once your instance is up and running, Log in to <http://localhost:9000> using System Administrator credentials:
 - Username: admin
 - Password: admin

SonarQube Server: Login

The screenshot shows the SonarQube login interface. At the top center is the Sonar logo, which consists of a red stylized 's' icon followed by the word 'sonar'. Below the logo is a light gray rectangular form with rounded corners. In the center of this form is a small blue icon resembling a sound wave or a signal. Below the icon, the text "Log in to SonarQube" is displayed in a bold, dark font. The form contains two input fields: the first is labeled "Login *" and contains the text "admin"; the second is labeled "Password *" and contains five black dots representing masked input. At the bottom of the form are two buttons: a white "Go back" button and a blue "Log in" button. At the very bottom of the page, there is a thin horizontal line followed by the text "SonarQube™ technology is powered by [SonarSource SA](#)" on the left and links for "LGPL v3", "Community", "Documentation", and "Plugins" on the right.

SonarQube Server: Create Project

The screenshot shows the 'Create Project' page of the SonarQube Server interface. At the top, there is a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. On the far right of the navigation bar are a help icon and a user profile icon.

The main content area has a heading 'How do you want to create your project?'. Below it, a text block asks if the user wants to benefit from SonarQube's features like repository import and Pull Request decoration, and suggests creating the project from a favorite DevOps platform.

Below this, a note says 'First, you need to set up a DevOps platform configuration.' followed by five 'Setup' buttons:

- Import from Azure DevOps
- Import from Bitbucket Cloud
- Import from Bitbucket Server
- Import from GitHub
- Import from GitLab

Further down, another note says 'Are you just testing or have an advanced use-case? Create a local project.' followed by a single 'Create a local project' button.

At the bottom left, there is a warning message in a yellow box: '⚠️ Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

At the very bottom, there is footer information: 'SonarQube™ technology is powered by SonarSource SA' with a link, and links for 'Community Edition v10.7 (96327) ACTIVE', 'LGPL v3', 'Community', 'Documentation', 'Plugins', and 'Web API'.

SonarQube Server: Create Project

The screenshot shows the 'Create a local project' dialog in SonarQube. The form fields are as follows:

- Project display name ***: unina-delivery (green border)
- Project key ***: unina-delivery (green border)
- Main branch name ***: main

Below the form, there is a note: "The name of your project's default branch [Learn More](#)". At the bottom are two buttons: "Cancel" and "Next".

Warning message: **⚠️ Embedded database should be used for evaluation purposes only**
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

At the bottom of the page, the footer includes:
SonarQube™ technology is powered by [SonarSource SA](#) Community Edition v10.7 (96327) ACTIVE | LGPL v3 | Documentation | Plugins | Web API

SonarQube Server: Analysis Methods

The screenshot shows the 'Analysis Method' configuration page in SonarQube. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. Below the navigation is a breadcrumb trail showing the project 'unina-delivery' and branch 'main'. The main content area has tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity, along with Project Settings and Project Information dropdowns. A prominent section titled 'Analysis Method' contains the heading 'How do you want to analyze your repository?'. It lists several options: 'With Jenkins', 'With GitHub Actions', 'With Bitbucket Pipelines', 'With GitLab CI', 'With Azure Pipelines', and 'Other CI'. A note below 'Other CI' states: 'SonarQube integrates with your workflow no matter which CI tool you're using.' Another section labeled 'Locally' provides instructions for testing or advanced use-cases. At the bottom, a warning message in a yellow box states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.' The footer contains links for SonarQube technology, Community Edition, LGPL v3, Community, Documentation, Plugins, and Web API.

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More ? A

unina-delivery / main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

- With Jenkins
- With GitHub Actions
- With Bitbucket Pipelines
- With GitLab CI
- With Azure Pipelines
- Other CI
SonarQube integrates with your workflow no matter which CI tool you're using.

Locally

Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.

⚠️ Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by [SonarSource SA](#)

Community Edition v10.7 (96327) ACTIVE [LGPL v3](#) [Community](#) [Documentation](#) [Plugins](#) [Web API](#)

SonarQube Server: Local Analysis

The screenshot shows the SonarQube interface for a project named "unina-delivery". The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search bar. Below the navigation is a breadcrumb trail: unina-delivery / main. The main content area has tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity, along with Project Settings and Project Information dropdowns.

The central section is titled "Analyze Method > Locally" and "Analyze your project". It displays a message: "We initialized your project on SonarQube, now it's up to you to launch analyses!"

The first step, "1 Provide a token", is highlighted. It features two buttons: "Generate a project token" (which is selected) and "Use existing token". Below these are fields for "Token name" (containing "Analyze \"unina-delivery\"") and "Expires in" (set to "No expiration"). A "Generate" button is also present. A note below the fields states: "Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#) for more information." A smaller note at the bottom of this section says: "The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#)".

The second step, "2 Run analysis on your project", is shown below the token section.

A warning message at the bottom left of the page reads: "⚠️ Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." Other footer links include "SonarQube™ technology is powered by [SonarSource SA](#)", "Community Edition v10.7 (96327) ACTIVE", "LGPL v3", "Community", "Documentation", "Plugins", and "Web API".

SonarQube Server: Local Analysis

The screenshot shows the SonarQube interface for a local analysis of the 'unina-delivery' project. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. Below the navigation is a breadcrumb trail showing the project name and branch ('main'). The main content area has tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity, along with Project Settings and Project Information dropdowns. A sub-header indicates the analysis method is 'Locally'. The central feature is the 'Analyze your project' section, which displays a token for analysis: 'Analyze "unina-delivery": sqp_10fd3b2da3f1664f4641ef283488e02ce27fb84d'. It includes a warning about the token's security and a 'Continue' button. Another step, 'Run analysis on your project', is shown below. A note at the bottom cautions against using the embedded database for evaluation purposes due to its limitations. The footer provides information about SonarQube's power source and various community and documentation links.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More ? A

unina-delivery / main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "unina-delivery": sqp_10fd3b2da3f1664f4641ef283488e02ce27fb84d ⚡

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

Continue

2 Run analysis on your project

⚠️ Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by [SonarSource SA](#) ↗ Community Edition v10.7 (96327) ACTIVE ↗ LGPL v3 ↗ Community ↗ Documentation ↗ Plugins ↗ Web API

SonarQube Server: Running an analysis

2 Run analysis on your project

What option best describes your project?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

Execute the Scanner for Maven

Running a SonarQube analysis with Maven is straightforward. You just need to run the following command in your project's folder.

```
mvn clean verify sonar:sonar \
-Dsonar.projectKey=unina-delivery \
-Dsonar.projectName='unina-delivery' \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.token=sq..._10fd3b2da3f1664f4641ef283488e02ce27fb84d
```

 Copy

Please visit the [official documentation of the Scanner for Maven](#) for more details.

If you're on Windows, run the above command using CMD.exe (PowerShell won't work)



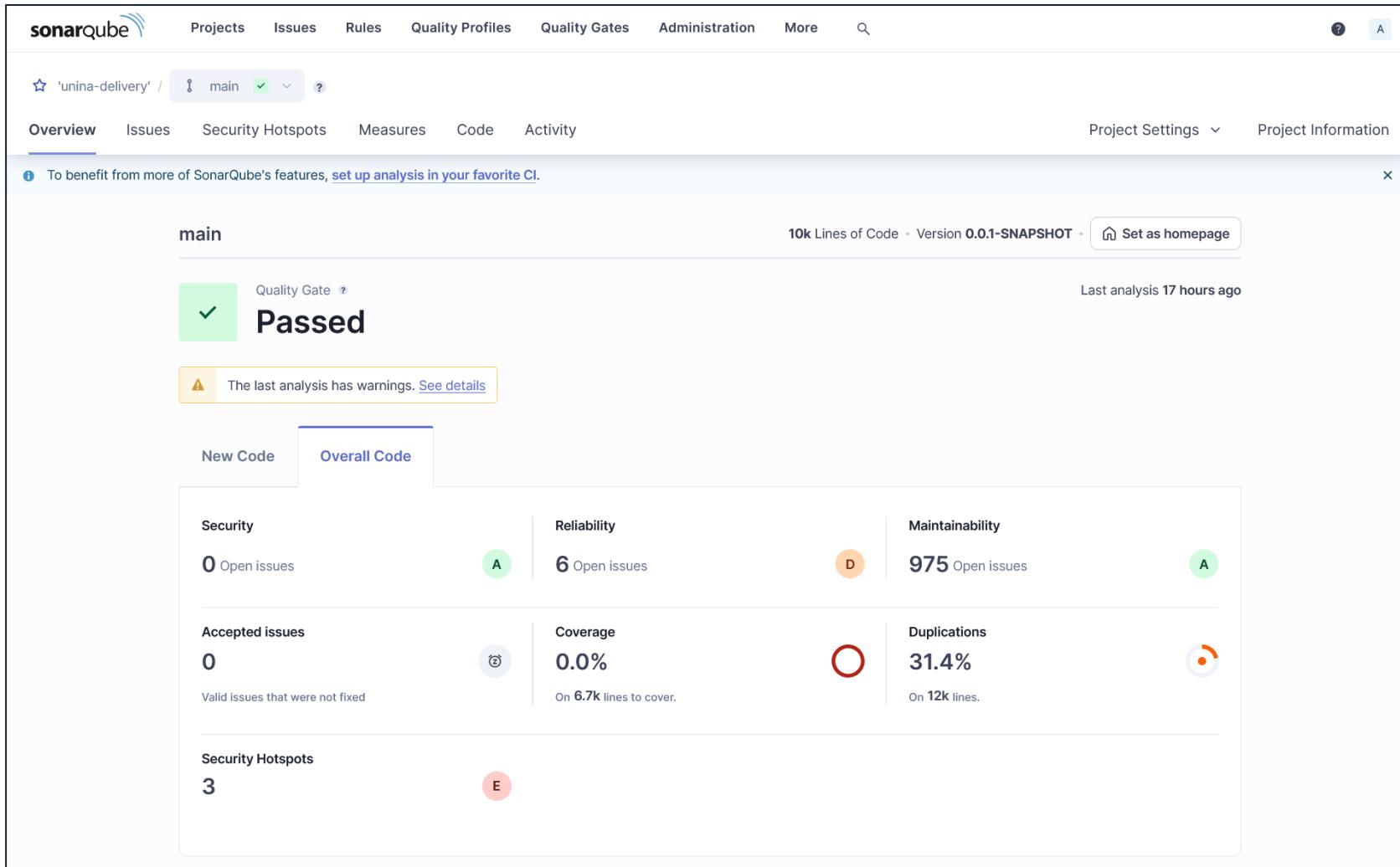
Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

While you're waiting, why not consider upgrading to our [Developer Edition](#)? It offers additional features such as [Branch Analysis](#) and [Pull Request Analysis](#).

SonarQube Server: Running an Analysis

```
@luigi → D/O/T/S/sonar $ docker run -d --name sonarqube  
-e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:10.7-community  
...  
[INFO] 16:55:01.681 CPD Executor CPD calculation finished (done) | time=97ms  
[INFO] 16:55:04.953 Analysis report generated in 229ms, dir size=1.3 MB  
[INFO] 16:55:05.118 Analysis report compressed in 139ms, zip size=414.5 kB  
[INFO] 16:55:05.169 Analysis report uploaded in 47ms  
[INFO] 16:55:05.175 ANALYSIS SUCCESSFUL, you can find the results at:  
      http://localhost:9000/dashboard?id=unina-delivery  
[INFO] 16:55:05.238 Analysis total time: 23.735 s  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 30.596 s  
[INFO] Finished at: 2024-11-16T16:55:05+01:00  
[INFO] -----
```

SonarQube Server: Project Dashboard



Maven Integration

- You don't need to manually execute the above command everytime you need to run an analysis
- You can include the Sonar analysis in the Maven build pipeline
 - Add the Sonarsource Maven Plugin to your POM
 - Add the sonar goal within the pipeline (e.g.: in the verify phase)
 - Then you can perform Sonar analyses by just running «mvn verify»

Modify Maven POM (1)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>org.example</groupId>
8      <artifactId>00BD_2122_1</artifactId>
9      <packaging>jar</packaging>
10     <version>1.0-SNAPSHOT</version>
11
12     <name>00BD_2122_01</name>
13
14     <properties>
15         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16         <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
17         <log4j2-version>2.19.0</log4j2-version>
18         <sonar.projectKey>INGSW_LECTURE_EXAMPLE</sonar.projectKey>
19         <sonar.login>sqp_a7672d91524e81baec9f017654b3649e94cfb98c</sonar.login>
20         <sonar.host>http://localhost:9000</sonar.host>
21     </properties>
```

Modify Maven POM (2)

```
84     <build>
85         <defaultGoal>install</defaultGoal>
86         <plugins>
87             <plugin>
88                 <groupId>org.sonarsource.scanner.maven</groupId>
89                 <artifactId>sonar-maven-plugin</artifactId>
90                 <version>3.9.1.2184</version>
91                 <executions>
92                     <execution>
93                         <goals>
94                             <goal>sonar</goal>
95                         </goals>
96                         <phase>verify</phase>
97                     </execution>
98                 </executions>
99             </plugin>
```

SonarQube Cloud

- SonarQube Cloud is SonarQube Server, offered as a SaaS in the cloud
- You don't need to bother running a local instance of SonarQube
- It's free for open-source projects
- If your project is in a public GitHub repository, you can effortlessly include it in SonarCloud
 1. Login into SonarCloud using your GitHub credentials (select free plan)
 2. Select the repository your project is in
 3. Enjoy a completely managed SonarQube instance in the cloud

A little digression on CI/CD

Delivering a Software Product

- When working on a software product, we want to produce high-quality software, and to deliver it **rapidly**
- **The job isn't done as soon as we finish writing the source code**
- The project needs to be built (e.g.: produce a JAR)
- We need to execute tests
- We need to run automated code inspections
- We need to **deploy** the new version (e.g.: install it whenever it needs to be installed)

Integration Challenges

- When large teams of developers are working on a project, integrating changes from multiple contributors becomes a challenge
- Even if each change worked in isolation, we do not know if all changes work when merged together!
- Integrating multiple weeks/months of work can be a cumbersome task
- Integration requires time and work, and additional issues might emerge
- Integrating every small changes with a higher frequency is a better approach and allows faster integrations

Continuous Integration (CI)

Continuous Integration is a software development practice where each member of a team merges their changes into a codebase together with their colleagues changes **at least daily**.

Key points:

- Use a shared repository (with a main branch)
- Have an automated build process in place (e.g.: Maven)
- Have automated tests in place
- Every push to the shared repository should trigger a build
 - Commits that break the build are not merged. The dev that pushed a breaking change gets notified (and can fix it work before pushing again)

Continuous Delivery / Deployment (CD)

- The aim of Continuous Delivery is that the product should always be in a state where it is possible to release the latest build. This is essentially ensuring that the release to production is only a business decision.
 - To ensure this, testing should be conducted automatically on every build
- This might be take a step further, and include automatic deployment to production of the latest non-breaking build that passed all tests and quality checks

SonarQube: Quality Gates

- In a CI/CD context, we may want to ensure that code adheres to established quality requirements
- With SonarQube, we can make a build fail based on quality requirements
- This is done with **Quality Gates**
 - A set of quality requirements that code (or new code) must satisfy
 - E.g.: no blocker or critical issues; at least 80% code coverage in tests, ...
 - A codebase either passes a quality gate or not

SonarQube: Quality Gates

The screenshot shows the SonarQube interface with the 'Quality Gates' tab selected. On the left, there's a sidebar with a 'Quality Gates' section containing a 'Create' button and a card for 'Sonar way' which is marked as 'DEFAULT BUILT-IN'. The main content area displays the 'Sonar way' quality gate configuration. It includes a note that it's the 'only quality gate you need to practice Clean as You Code'. The 'Conditions' section lists four rules: 'New code has 0 issues', 'All new security hotspots are reviewed', 'New code is sufficiently covered by test' (with a coverage threshold of '80.0%'), and 'New code has limited duplication' (with a duplication threshold of '3.0%'). Below this, the 'Projects' section states that every project not specifically associated to a quality gate will be associated to this one by default.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More ? A

Quality Gates ? Create

Sonar way DEFAULT BUILT-IN

Copy

The only quality gate you need to practice Clean as You Code

Conditions

Your new code will be clean if: ?

New code has 0 issues

All new security hotspots are reviewed

New code is sufficiently covered by test Coverage is greater than or equal to 80.0% ?

New code has limited duplication Duplicated Lines (%) is less than or equal to 3.0% ?

Projects ?

Every project not specifically associated to a quality gate will be associated to this one by default.

SonarQube: Custom Quality Gates

- You can also define custom quality gates to enforce

Add Condition

Where?

On New Code

On Overall Code

Quality Gate fails when

Cyclomatic Complexity

Operator	Value
is greater than	15

Add Condition **Close**

SonarQube and CI/CD

- SonarQube can be integrated in CI/CD pipelines to ensure that code quality does not degrade
- E.g.: refuse merges which do not pass a quality gate

READINGS AND REFERENCES

- SonarSource website
<https://www.sonarsource.com/>
- SonarQube docs
<https://docs.sonarsource.com/sonarqube/latest/>
- Continuous Integration, Martin Fowler
<https://martinfowler.com/articles/continuousIntegration.html>
- Automatic Assessment of Architectural Anti-patterns and Code Smells in Student Software Projects, L. L. L. Starace (among other authors)
<https://dl.acm.org/doi/pdf/10.1145/3661167.3661290>