

45426: Teste e Qualidade de Software

Code improvement: refactoring and static code analysis

Ilídio Oliveira

v2022-04-19



Learning objectives

Identify the occurrence of “bad smells” in code

Propose refactoring options for given “code smells”

Explain the role of Inspectors (static code analysis)

Describe the metrics used in SonarQube

Define the concept of technical debt and explain how it should be managed in a SQEnvironment

THE REAL CODER

geek & poke



THERE'S A METHOD CALLED
`getSerialNumber()`
IT HAS THE COMMENT
`// get the serial number`
WHAT DO YOU THINK IT COULD
DO?



geek & poke



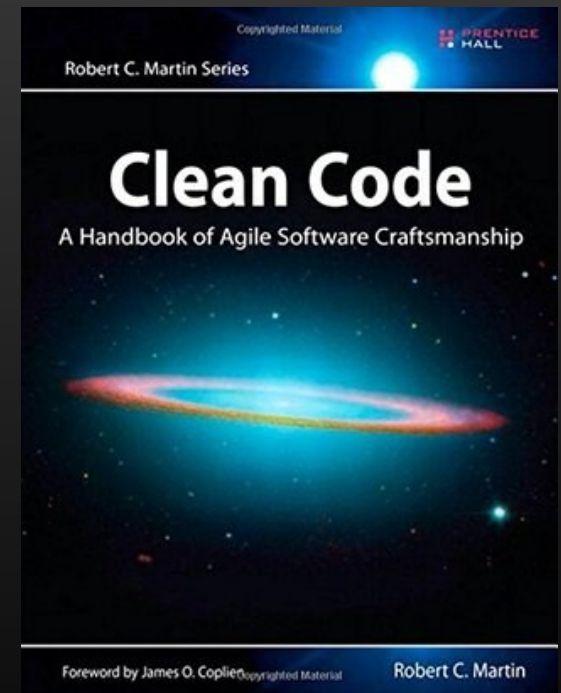
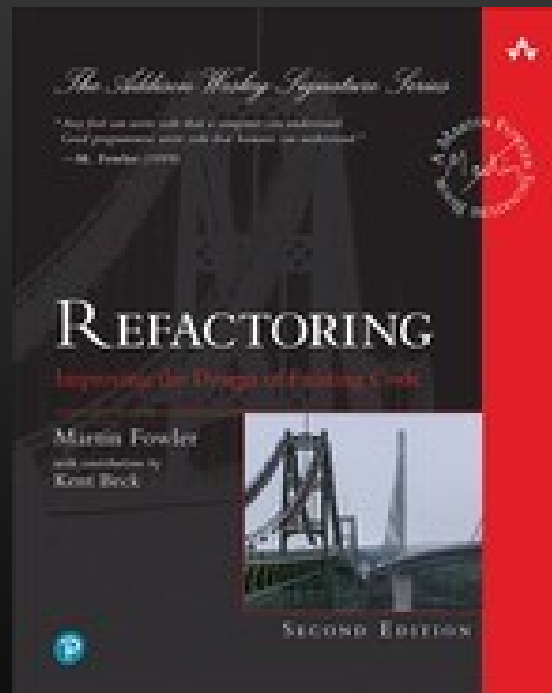
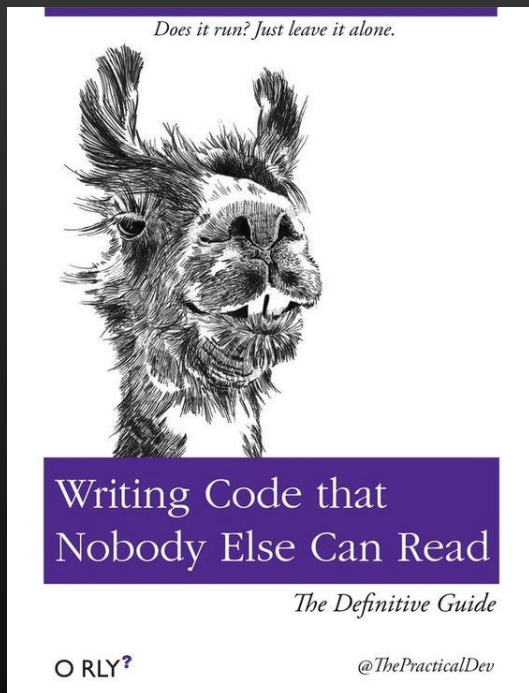
I'LL LOOK
INTO THE
CODE

ANYTHING?



"ONLY IN CODE WE TRUST"

Not all code is equally easy to maintain



Find the intruder...

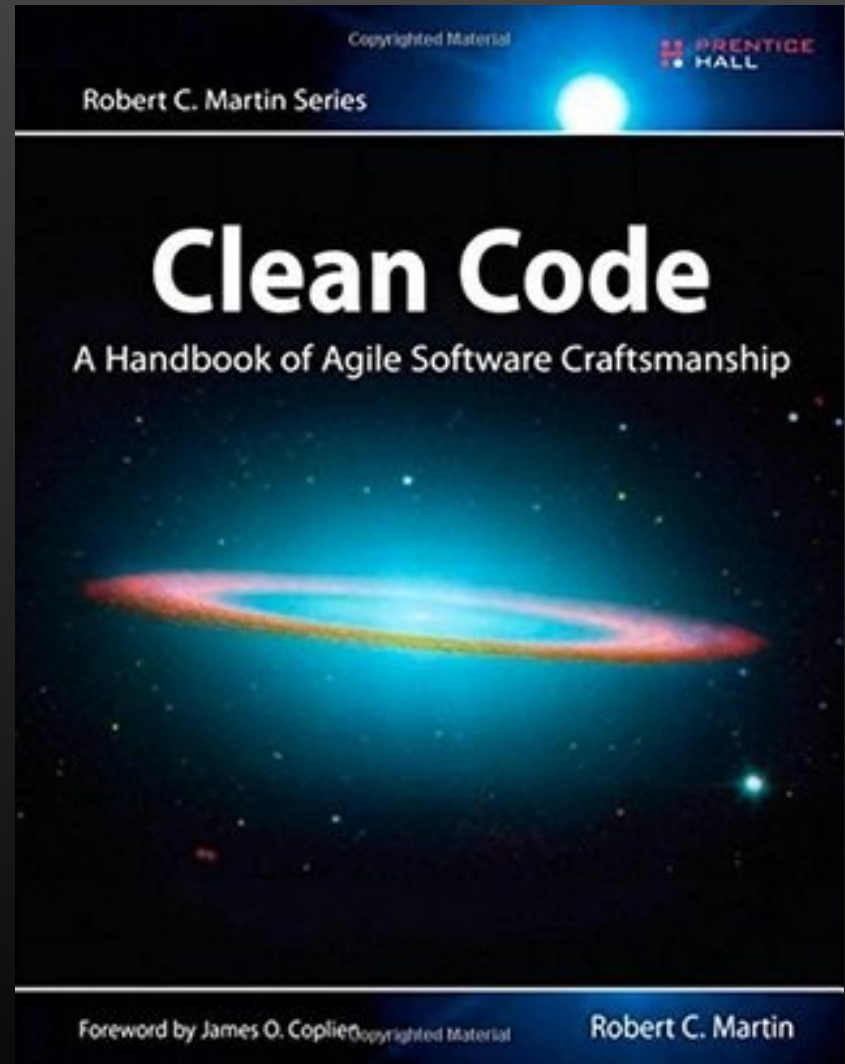
Clean code

Code as communication

- code is read much more often than it is written!
- anyone can write code that compiles, but...
- true mastery is to **write code for other people** (easy to understand and maintain)

Video talks by “uncle Bob”

6 talks on Clean Code (see link in Moodle)



<https://learning.oreilly.com/library/view/clean-code-a/9780136083238/>

The name of a variable, function, or class, should answer all the big questions. It should tell you why it exists, what it does, and how it is used. If a name requires a comment, then the name does not reveal its intent.

```
int d; // elapsed time in days
```

The name `d` reveals nothing. It does not evoke a sense of elapsed time, nor of days. We should choose a name that specifies what is being measured and the unit of that measurement:

```
int elapsedTimeInDays;  
int daysSinceCreation;  
int daysSinceModification;  
int fileAgeInDays;
```

Choosing names that reveal intent can make it much easier to understand code. What is the purpose of this code?

```
public List<int[]> getThem() {  
    List<int[]> list1 = new ArrayList<int[]>();  
    for (int[] x : theList)  
        if (x[0] == 4)  
            list1.add(x);  
    return list1;  
}
```

Chapter 2: Meaningful Names

Introduction

Use Intention-Revealing Names

Avoid Disinformation

Make Meaningful Distinctions

Use Pronounceable Names

Use Searchable Names

► Avoid Encodings

Avoid Mental Mapping

Class Names

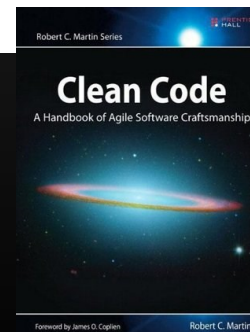
Method Names

Don't Be Cute

Pick One Word per Concept

Don't Pun

Use Solution Domain Names



Explain Yourself in Code

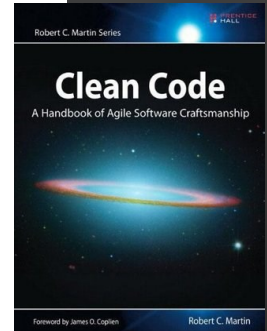
There are certainly times when code makes a poor vehicle for explanation. Unfortunately, many programmers have taken this to mean that code is seldom, if ever, a good means for explanation. This is patently false. Which would you rather see? This:

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) &&
    (employee.age > 65))
```

Or this?

```
if (employee.isEligibleForFullBenefits())
```

It takes only a few seconds of thought to explain most of your intent in code. In many cases it's simply a matter of creating a function that says the same thing as the comment you want to write.



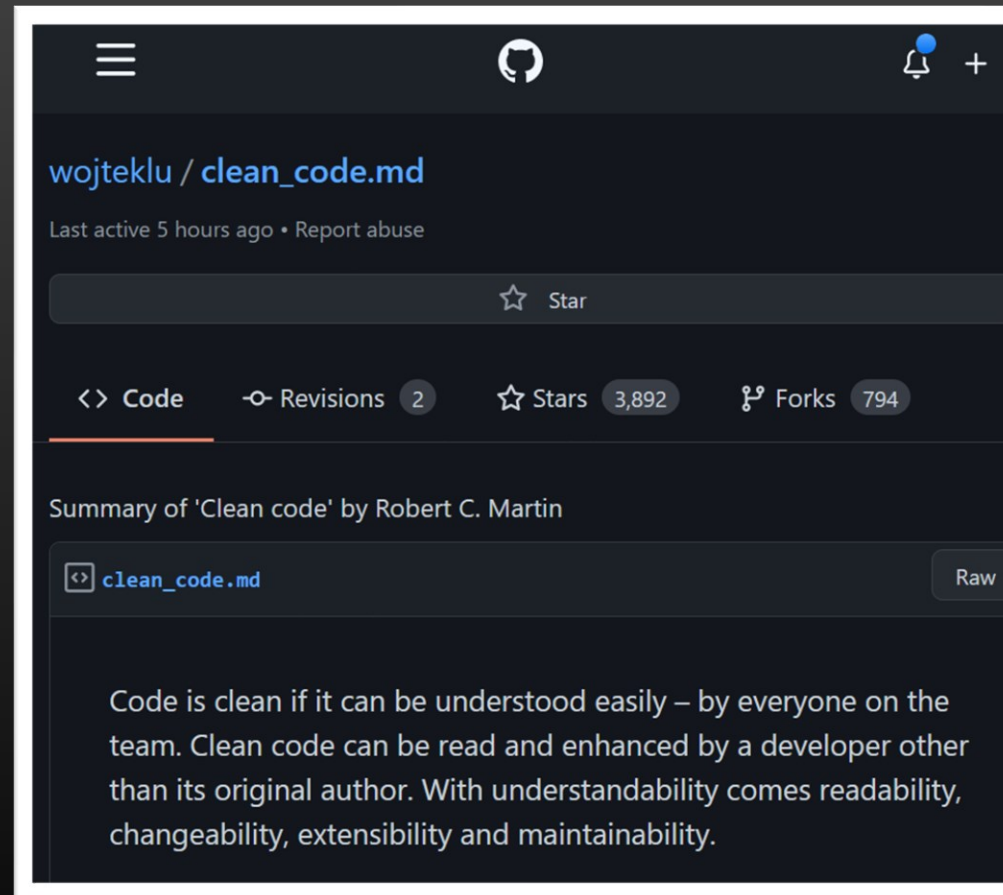
SucheG commented on Apr 27, 2021

at school: "Comments are important to clarify what you code do"
at book: "Comments are sign that yor code is bad designed"

In: <https://gist.github.com/wojtekl/73c6914cc446146b8b533c0988cf8d29>

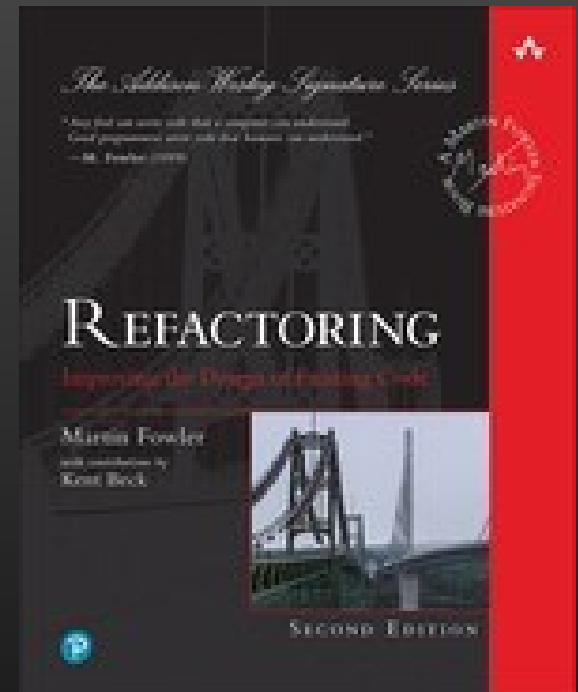
Many “articles” inspired in clean coding

Summaries on “Clean Code” available from Internet



<https://gist.github.com/wojtekl/73c6914cc446146b8b533c0988cf8d29>

Refactoring practices



Code refactoring

Refactoring is a controlled technique for improving the design of an existing code base

...altering its internal structure without changing its external behavior.

Key aspects:

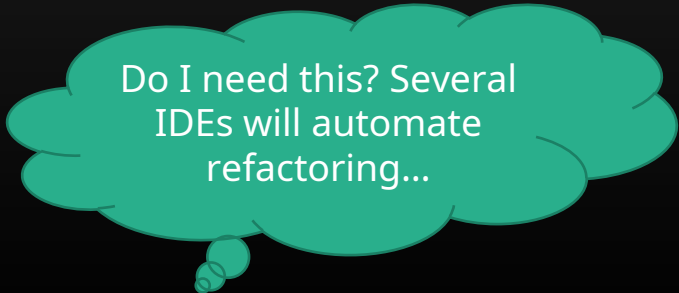
- series of “small” transformations
- preserving functionality & correctness.

Examples

- Extract (duplicate code into a) method
- Extract interface (segregate responsibilities)

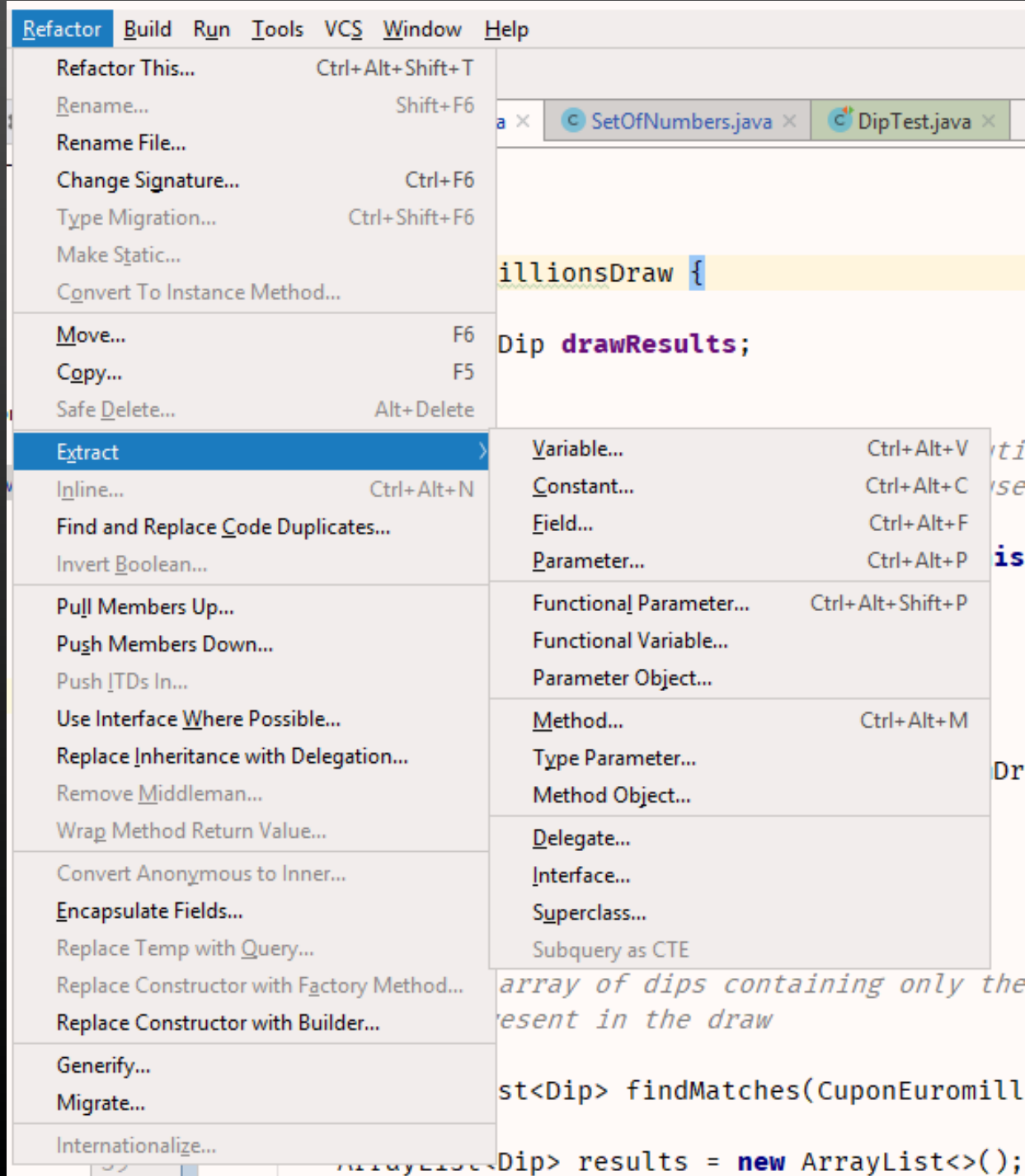
See also:

- Source Making: [refactoring techniques](#)
- Fowler: [Catalog of refactoring situations](#)



Do I need this? Several IDEs will automate refactoring...

IntelliJ support



When to refactor?

Resolve “code smells” (anti-patterns)

See: catalog of [code smells](#)

Examples:

Duplicate code → Extract method

Long method → Extract method

Feature Envy → Move method

Long parameters list → introduce object / preserve object

Picking a *mantra*...

'Maintainable code,
maintainable code,
maintainable code!'

'Test, test, test': WHO chief's coronavirus message to world

By Emma Farge, John Revill

3 MIN READ



LAUSANNE/ZURICH (Reuters) - The World Health Organization called on all countries on Monday to ramp up their testing programs as the best way to slow the advance of the coronavirus pandemic, and also urged companies to boost production of vital equipment to overcome acute shortages.



Why refactoring? MAINTAIN & EVOLVE!

Cleaner code

is easier to understand and maintain

Better design

adjust to the current understanding of the architecture

Reduce complexity

easier to understand and evolve

Make the code more reusable

component-like thinking (generalize for other needs)

Improve performance

Improve security

by removing vulnerabilities



Code inspection

**Analysis of code patterns,
without running the code**

**Examples of issues found
in SA:**

*Referencing a variable with an
undefined value*

Variables that are never used

Unreachable (dead) code

*Programming standards
violations*

Security vulnerabilities

- *E.g.: possible SQL injections*

Internationalization (i18n) issues



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Current events](#)
[Random article](#)
[About Wikipedia](#)
[Contact us](#)
[Donate](#)

[Contribute](#)
[Help](#)

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

Article

Talk

Read

Edit

View history

Search Wikipedia



lint (software)

From Wikipedia, the free encyclopedia

lint, or a **linter**, is a [static code analysis](#) tool used to flag programming errors, [bugs](#), stylistic errors and suspicious constructs.^[4] The term originates from a [Unix utility](#) that examined [C language](#) source code.^[1]

Contents [\[hide\]](#)

lint

Original author(s)	Stephen C. Johnson
Developer(s)	AT&T Bell Laboratories
Initial release	July 26, 1978; 42 years ago ^[1]
Written in	C
Operating system	Cross-platform
Available in	English
Type	Static program analysis tools

Catalogs of code weaknesses (setting the vocabulary)



A Community-Developed List of Software & Hardware Weakness Types



ID Lookup: Go

[Home](#) | [About](#) | [CWE List](#) | [Scoring Search](#) | [Community](#) | [News](#) | [Guidance](#)

CWE™ is a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

View the List of Weaknesses

by Software Development

by Hardware Design

by Research Concepts

Search CWE

Easily find a specific software or hardware weakness by performing a search of the CWE List by keywords(s) or by CWE-ID Number. To search by multiple keywords, separate each by a space.

ENHANCED BY Google



NPE due to a bad exception handling

```
// Execute
Process process = null;
try{
    if(cmd.length == 1) {
        process = Runtime.getRuntime().exec( cmd[0] );
    } else {
        process = Runtime.getRuntime().exec( cmd );
    }
}
catch(Exception e){
    e.printStackTrace();
}

try {
    if(inputToStdIn) {
        sendInput(process, stream);
    } else {
        process.getOutputStream().close();
    }
}
```

'process' is by definition null here.

**If an exception is thrown when
executing the command line,
'process' remains null.**

**So a NullPointerException will be
thrown later.**

NullPointerException might be thrown as 'process' is nullable here ...

2 months ago ▾ L193

🚫 Blocker ○ Open Not assigned Not planned 10min debt

🔍 bug, cert, cwe, owasp-a1, owasp-a2, owasp-a6, security

}

<https://blog.sonarsource.com/sonaranalyzer-for-java-tricky-bugs-are-running-scared/>

Useless condition

```
// Handle web socket routes
if (websocketServletContextHandler == null) {
    server.setHandler(handler);
} else {
    List<Handler> handlersInList = new ArrayList<>();
    handlersInList.add(handler);

    // WebSocket handler must be the last one
    if (websocketServletContextHandler != null) {
```

If 'websocketServletContextHandler' is null in this branch, it can't be nullable in the 'else' branch

Change this condition so that it does not always evaluate to "true"

2 months ago L115

Blocker Open Not assigned Not planned 15min debt

bug, cwe, misra

```
        handlersInList.add(websocketServletContextHandler);
    }

    HandlerList handlers = new HandlerList();
    handlers.setHandlers(handlersInList.toArray(new Handler[handlersInList.size()]));
    server.setHandler(handlers);
}
```

<https://blog.sonarsource.com/sonaranalyzer-for-java-tricky-bugs-are-running-scared/>

Suspect unreachable branch

```
TemporaryResources tmp = new TemporaryResources();
File output = null;
try {
    TikaInputStream tikaStream = TikaInputStream.get(stream, tmp);
    File input = tikaStream.getFile();
    String cmdOutput = computePoT(input);
    FileInputStream ofStream = new FileInputStream(new File(
        input.getAbsolutePath() + ".of.txt"));
    FileInputStream ogStream = new FileInputStream(new File(
        input.getAbsolutePath() + ".hog.txt"));
    extractHeaderOutput(ofStream, metadata, "of");
    extractHeaderOutput(ogStream, metadata, "og");
    xhtml.startDocument();
    doExtract(ofStream, xhtml, "Histogram of Optical Flows (HOF)",
        metadata.get("of_frames"), metadata.get("of_vecSize"));
    doExtract(ogStream, xhtml, "Histogram of Oriented Gradients (HOG)",
        metadata.get("og_frames"), metadata.get("og_vecSize"));
    xhtml.endDocument();
} finally {
    tmp.dispose();
    if (output != null) {
```

'output' is in fact never initialised so indeed always null so the content of the branch is unreachable.

Change this condition so that it does not always evaluate to "false" ...

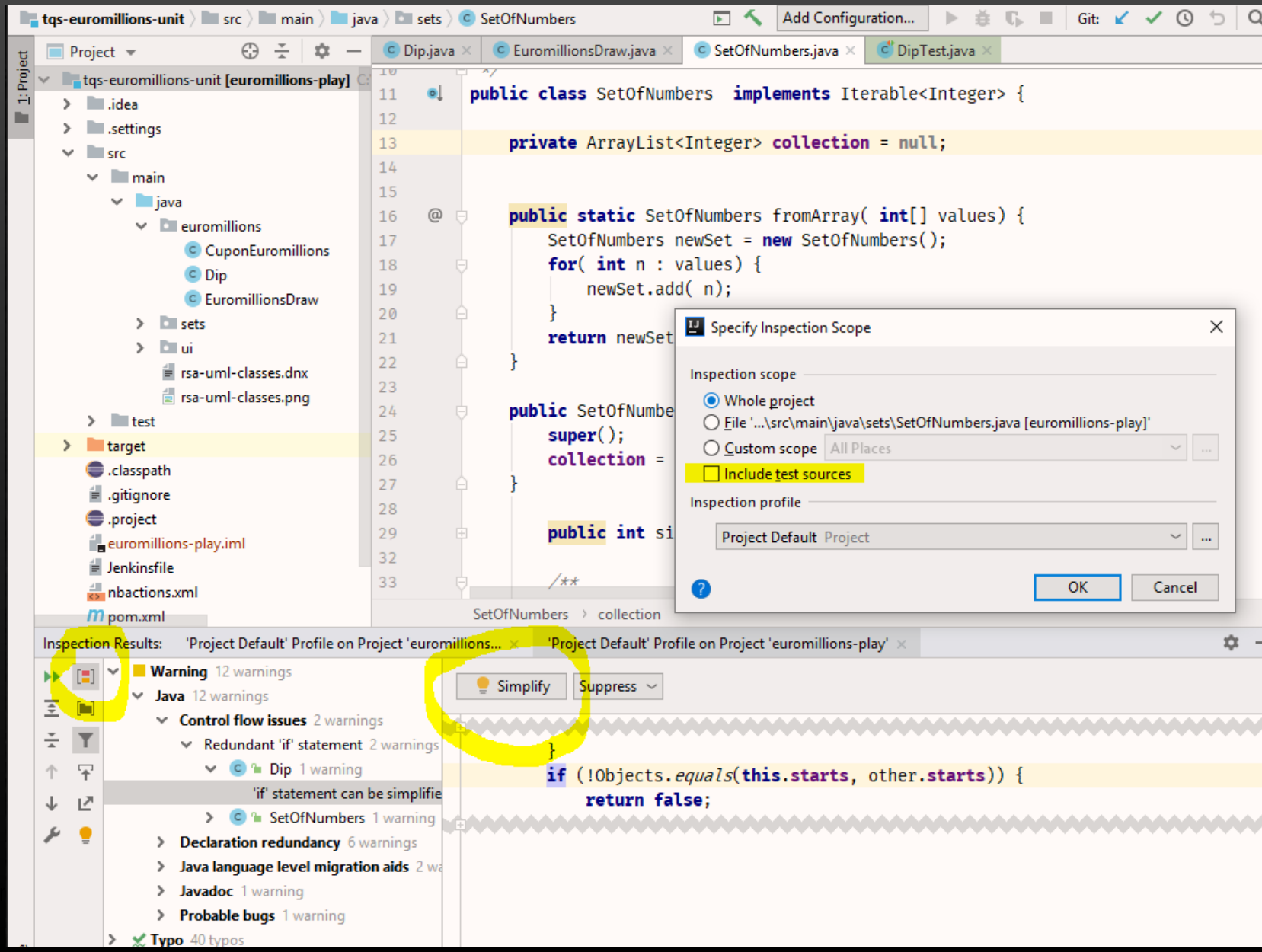
4 months ago ▾ L145 🔍 📄

🚫 Blocker ○ Open Not assigned Not planned 15min debt

🐞 bug, cwe, misra

```
        output.delete();
    }
}
```

Code inspection in IntelliJ



Advanced inspection frameworks



Code Smells



Bugs



Vulnerabilities



C O D A C Y

Product

Automate your code quality

Automatically identify issues through static code review analysis. Get notified on security issues, code coverage, code duplication, and code complexity in every commit and pull request, directly from your current workflow.

Sonar Qube concepts

Code Smell

A maintainability-related issue in the code. Leaving it as-is means that at best maintainers will have a harder time than they should making changes to the code. At worst, they'll be so confused by the state of the code that they'll introduce additional errors as they make changes.

Bug

An issue that represents something wrong in the code. If this has not broken yet, it will, and probably at the worst possible moment. This needs to be fixed. Yesterday.

Vulnerability

A security-related issue which represents a potential backdoor for attackers.



<https://docs.sonarqube.org/display/SONAR/Concepts>

Quality gates

Ready for delivery? Yes, if the defined Quality Gate is met.

Recommended Quality Gate

Metric	Over Leak Period	Operator	Warning	Error
Coverage on New Code	Always	is less than ▼	<input type="text"/>	80
Duplicated Lines on New Code (%)	Always	is greater than ▼	<input type="text"/>	3
Maintainability Rating on New Code	Always	is worse than	<input type="text"/>	A × ▼
Reliability Rating on New Code	Always	is worse than	<input type="text"/>	A × ▼
Security Rating on New Code	Always	is worse than	<input type="text"/>	A × ▼



- Architecture and Integration
- › Requirements
- › Setup and Upgrade
- › Analyzing Source Code
- ▼ User Guide
 - Fixing the Water Leak
 - Quality Gates
 - › Projects
 - › Issues
 - › Rules
 - Built-in Rule Tags
 - User Account
 - User Token
 - › Code Viewer
 - UI Tips
 - › **Metric Definitions**
 - Concepts
 - Activity and History
 - Visualizations
 - › Project Administration Guide
 - › Administration Guide
 - Documentation for previous versions

Metric Definitions

Created by Anonymous on Jan 30, 2018

Table of Contents

- Complexity
- Duplications
- Issues
- Maintainability
- Quality Gates
- Reliability
- Security
- Size
- Tests

<https://docs.sonarqube.org/latest/user-guide/metric-definitions/>

This is not an exhaustive list of metrics. For the full list, consult the *api/metrics* WebAPI on your SonarQube instance.

Complexity

Name	Key	Description
Complexity	complexity	It is the complexity calculated based on the number of paths through the code. Whenever the control flow of a function splits, the complexity counter gets incremented by one. Each function has a minimum complexity of 1. This calculation varies slightly by language because keywords and functionalities do. More details
Cognitive Complexity	cognitive_complexity	How hard it is to understand the code's control flow. See https://www.sonarsource.com/resources/wh

Complexity metric

cyclomatic complexity

defines the number of independent paths in the basis set of a program and provides you with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once

See also:

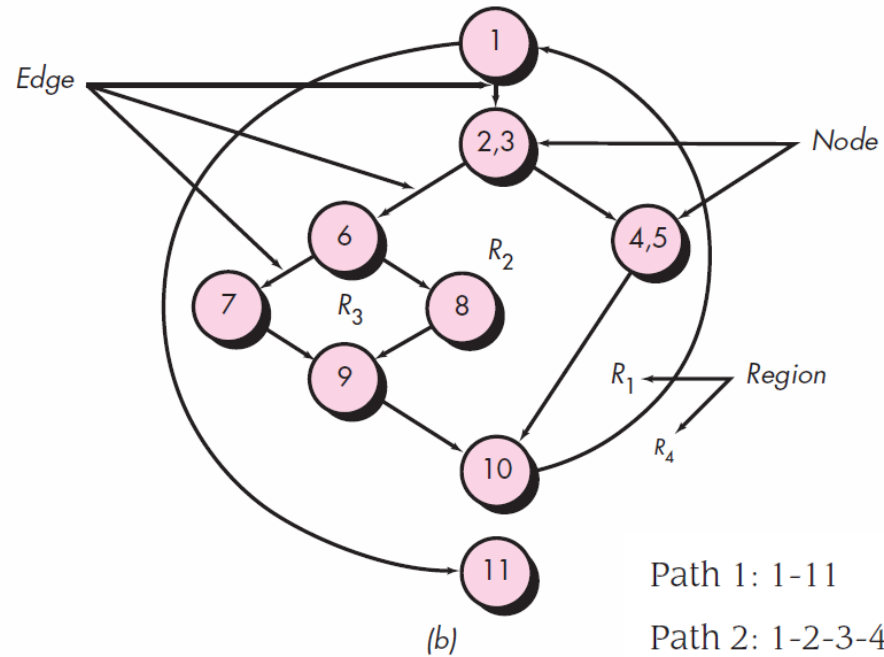
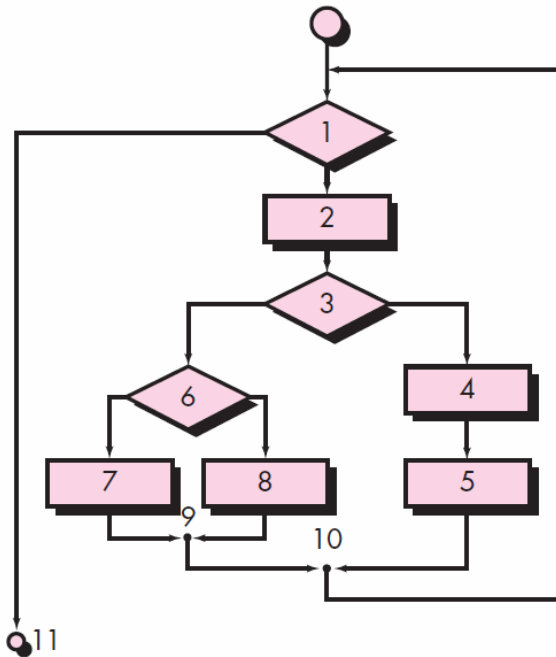
- Sonar [whitepaper](#)



A new way of measuring understandability

By G. Ann Campbell, SonarSource SA

FIGURE 18.2 (a) Flowchart and (b) flow graph



Path 1: 1-11

Path 2: 1-2-3-4-5-10-1-11

Path 3: 1-2-3-6-8-9-10-1-11

Path 4: 1-2-3-6-7-9-10-1-11

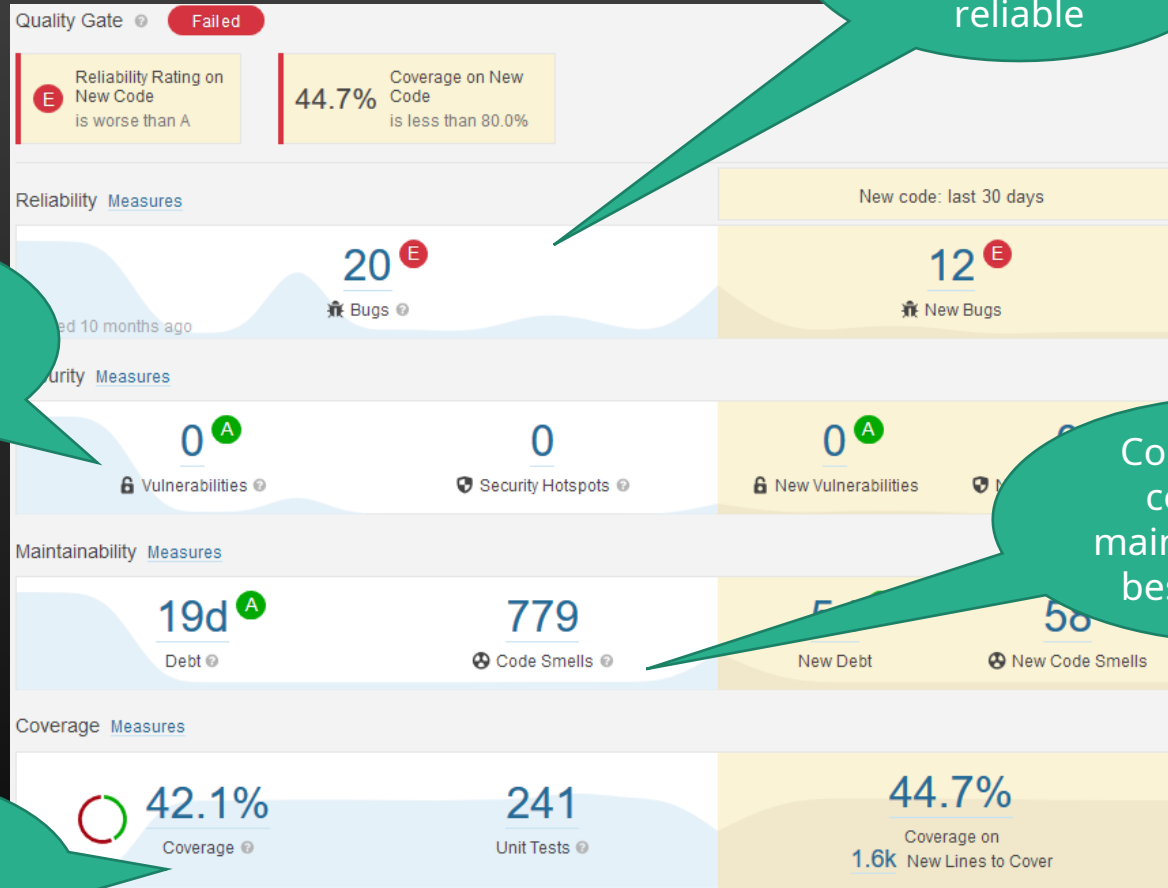
Cyclomatic complexity $V(G)$ for a flow graph G is defined as

$$V(G) = E - N + 2$$

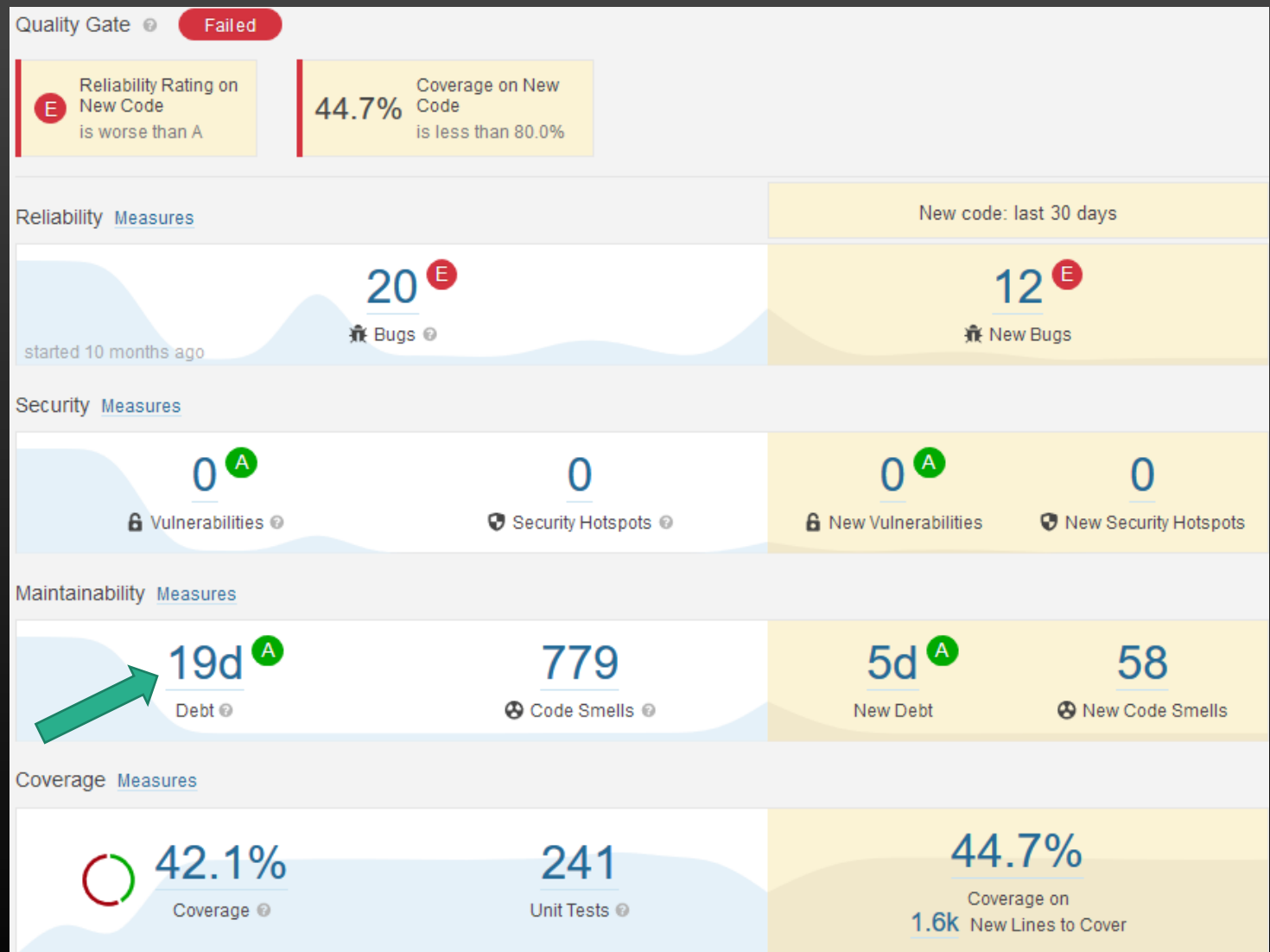
where E is the number of flow graph edges and N is the number of flow graph nodes.

$$V(G) = 11 \text{ edges} - 9 \text{ nodes} + 2 = 4.$$

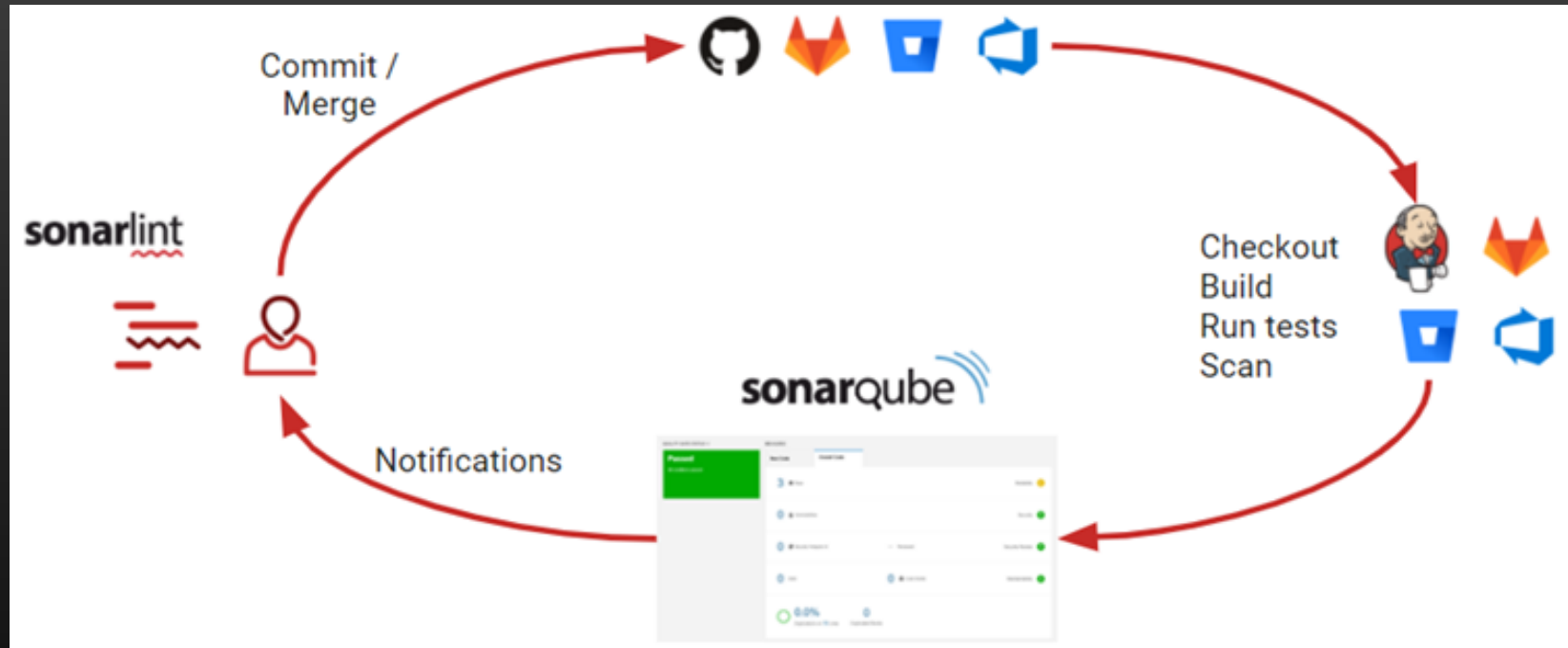
Sonar metrics for a QA dashboard



Technical debt



Integrate SonarQube in the development pipeline



References

Clean Code

- R. Martin, “Clean Code”,
- Suggested practices: <https://github.com/tum-esi/common-coding-conventions>