SonarQube

Venkat Ballem
Mobile Architect
https://www.dotcraft.tech

Timestamp: April 2020

Agenda

- Challenges
- SonarQube Intro
- Pull Request Analysis
- Developer Workflow
- Quality Profiles
- Quality Gates
- Test Coverage
- 3rd Party Plugin Support
- Limitations

Challenges:

Only **1** Challenge

How to write & maintain good

Quality Code

brand new team: code quality ????

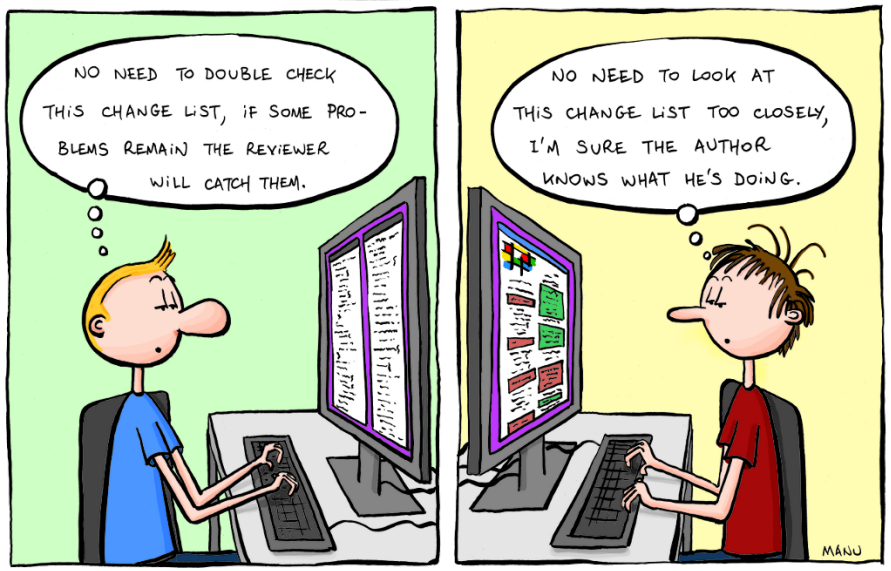
Code Review:

Below are some practical day-2-day challenges that we face besides suggesting best code-review guidelines…

Challenges:

No One Owns Quality:

- I didn't write this code; I don't know anything about this to review
- This is written by THAT guy, let him fix as he knows better
- I am not the best person to review PR



Lack of Interest/Time to Review:

- I am super busy right now; can u ask someone to review.
- mutual negotiation: **You approve mine <-> I approve yours** , done done!!

Lack of Skills/Experience:

- Some guys do review up-to their knowledge which sometimes not good enough
- New to Organization or Project or Process

Building Tech debts slowly:

- lack of knowledge on Code Smells, Design Principles
- miscellaneous compromise calls on quality

Aftermath of Each Merge:

Tech Debts – The Devil that travels with us

When it Happens?

- *No Proper feedback at right time , right place on code quality [* ***MAJOR CHALLENGE****]*
- Feedback is coming late in process
- Ineffective code reviews
- Compromise on Quality for multiple reasons
- Lack of foresight

What can Happen?

- It can simply shutdown the project
- More Tech debts --> More time & money
- No Money -> No Project

Technical Debt ≈ Financial Debts

Meet SonarQube:

- its approach is…

Clean as you Code

Clean as You Code means focusing on New Code for maximum Code Quality impact with minimum investment.

SonarQube:

- It ensures that the

Code

you write

Today

is

Clean, Safe & Solid

Major Axes of Quality Model

Reliability:

- **Bugs** track code that is demonstrably wrong or highly likely to yield unexpected behaviour.

Security:

- **Vulnerabilities** are raised on code that can be exploited by hackers.
- **Security Hotspots** are raised on security-sensitive code that requires manual review to assess whether or not a vulnerability exists.

Maintainability:

- **Code Smells** indicate weaknesses in design that may be slowing down development or increasing the risk of bugs or failures in the future. Bad code smells can be an indicator of factors that contribute to technical debt

Besides above 3 , it also measures below:

- **Code Coverage**
- **Duplicate Code**
- **Complexity**

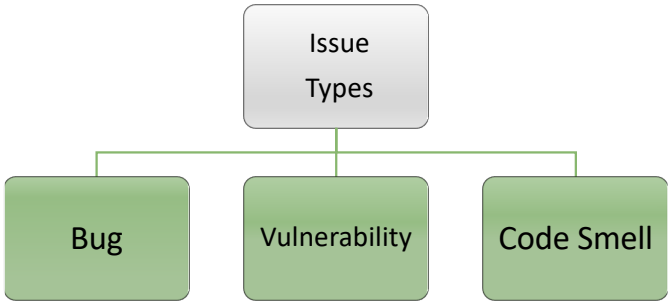
Old 7 Axes of Quality(5.6):

- Potential bugs
- Coding rules
- Tests
- Duplications
- Comments
- Architecture and design
- Complexity

Issue Types & Severity:

Issue:

- When a piece of code does not comply with a **rule**, an issue is logged on the **snapshot**.
- An issue can be logged on a **source file** or a **unit test** file.

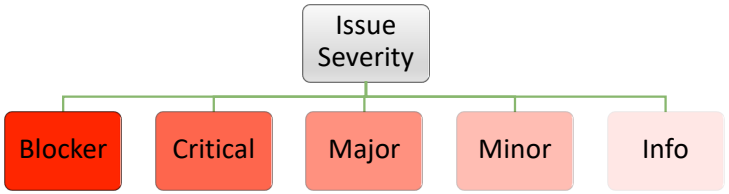


Issue Types:

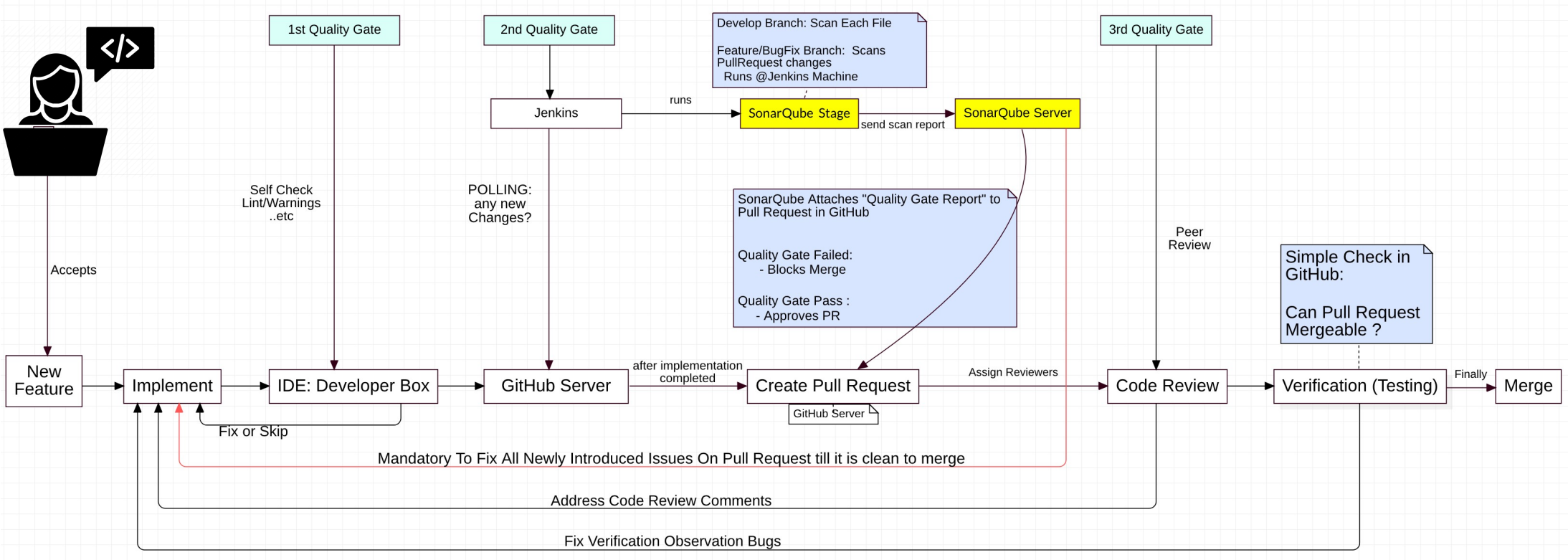
- Bugs
- Code Smells
- Vulnerabilities

Issue Severity:

- Blocker
- Critical
- Major
- Minor
- Info



Developer Workflow @ SonarQube + Jenkins + GitHub

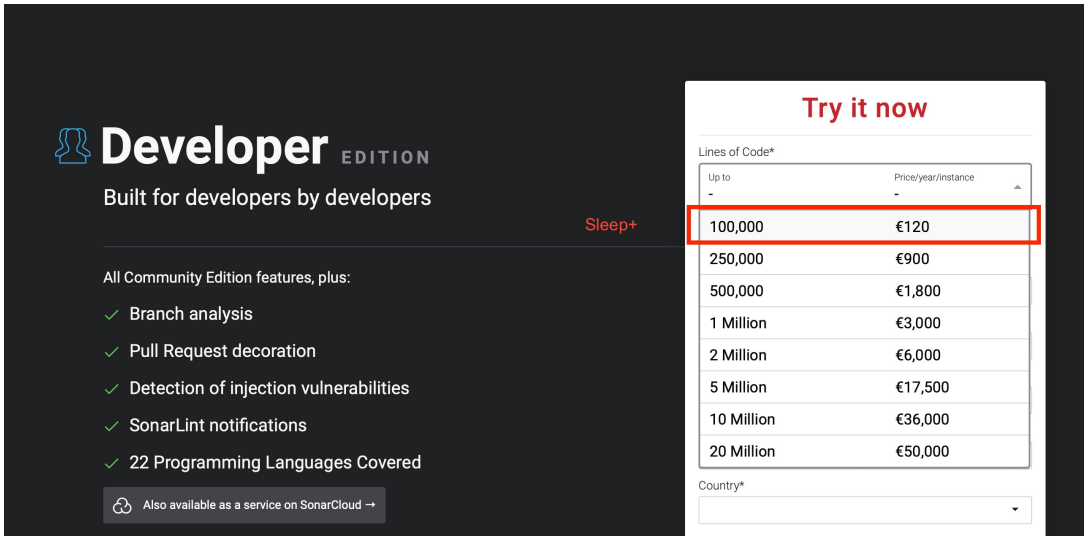


<u>Projects with out SonarQube PR Decoration</u>:

- Not Quick enough to get feedback on New Code
- Less Rules to ensure Code Quality on New Code
- possible slippage on code smells which leads to Tech Debts later
- Invest Additional Time & Money to clean later

<u>Projects with SonarQube PR Decoration</u>:

- Quick Feedback
- Empowering Developers to own code quality every day
- No look back on Code Quality later . CLEAN-AS-YOU-CODE
- Single Quality Gate which ensures Code is Clean before merge
- No Tech Debts in future.



Pull Request Analysis:

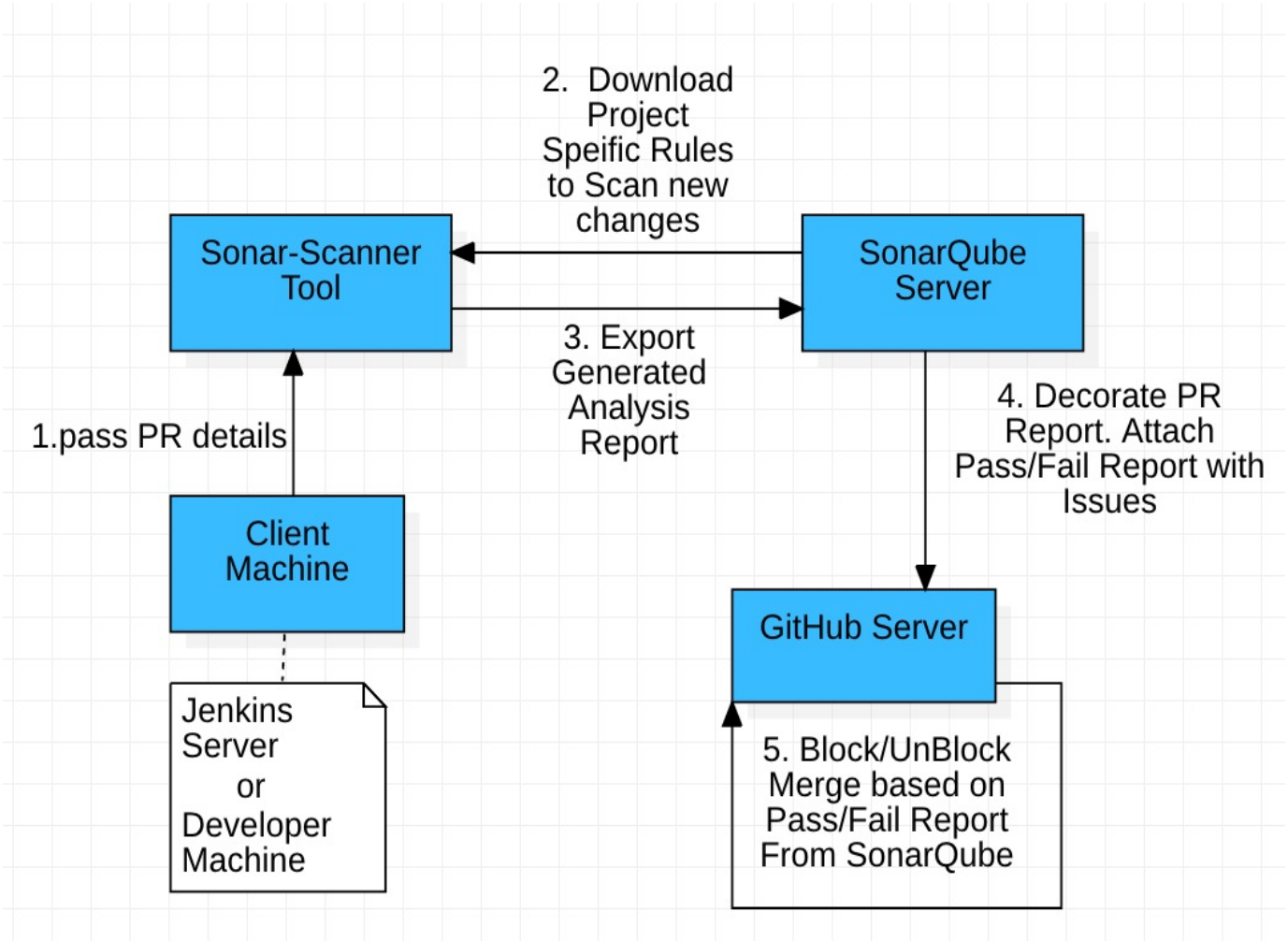
- **Allows :**
  - Analysis happens at Client Machine for faster scan
  - To see Pull Request Quality Gate status in the SonarQube UI.
  - Automatically decorate Pull Request with collected issues in GitHub interface directly

- **Dedicated Quality Gate for Pull Request:**
  - It uses your project's quality gate conditions that apply to "on New Code" metrics.
  - Each PR shows a quality gate status reflecting whether it Passed (green) or Failed (red).
  - PR analyses on SonarQube are deleted automatically after 30 days with no analysis

Notes:

- Pull Request Decoration is only available from **Developer Edition** which is not free
- Pull Request Decoration is NOW available for Community Edition as well with below free plugin
- https://github.com/mc1arke/sonarqube-community-branch-plugin
- This only works with Github for now.

Pull Request Decoration:



sonar.pullrequest.branch = feature/Parse-Login

sonar.pullrequest.key = 145

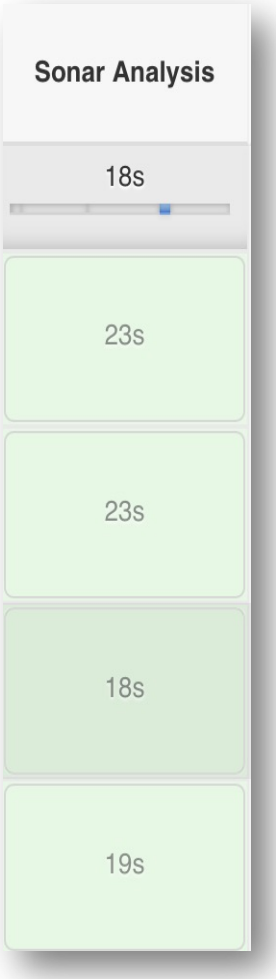
sonar.pullrequest.base. = develop

Scanning Performance:

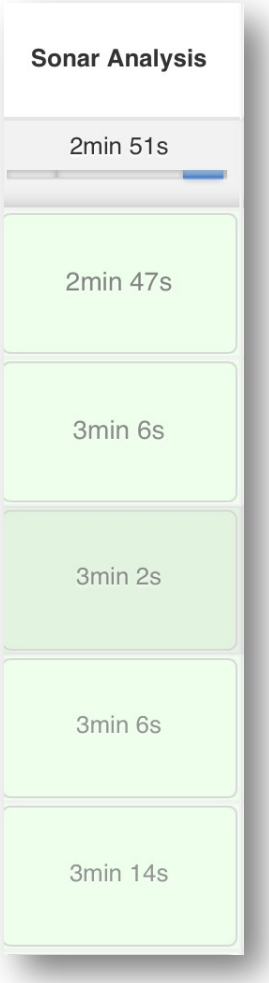
Less Time, Solid Feedback

Android LoC: 23k
iOS LoC: 23k

iOS



Android



Quality Profiles:

- Quality Profiles are collections of **Rules** to apply during an analysis.
    - **Rule** – A coding standard or practice which should be followed. Not complying with coding rules leads to Bugs, Vulnerabilities, Security Hotspots, and Code Smells. Rules can check quality on code files or unit tests.
- Can combine many different plugin rules under one profile
- Deactivate rules which are not appropriate to the the project specific language
- Provides inbuilt rules for Find-Bugs, Check-Style and PMD for Java Projects

Quality Gates:

- A Place where we enable threshold values on each metric that we want to track
- It ensures that the project must meet before it can be released into production
- It helps to know immediately whether new code is production-ready
- Allows to set Different threshold values on New Code (PR)
- Allows to set Baseline threshold values on Old Code

Demo-Project

Conditions on Overall Code

Metric	Operator	Value
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Rating	is worse than	A
Duplicated Lines (%)	is greater than	3.0%

Legacy Code :

Who Owns Legacy Code Quality?

- Team

Who Owns New Code Quality?

- The Respective Developer
- Developer is making sure that the new code is clean

How to Proceed with Old Code:

- **Quality Gate** helps here to create Baseline policy for the existing code
- It allows to define strict gates on selected metrics

How to Clean Old Code:

- Actively Plan and Clean
- Even without active clean-up, in the normal course of business the code base will gradually be cleaned up anyway as developers touch old code to make new changes.
- Areas of code that are modified frequently will be fixed quickly, making future maintenance of those high-traffic areas easier, cheaper, and more reliable.
- Less-trafficked areas of code will be cleaned up more slowly

Test Coverage

Sonar Analysers do not run our tests or generate reports. They only import pre-generated reports like other tools.

C/C++/Objective-C:

- sonar.cfamily.gcov.reportsPath = path to *.gcov reports
- sonar.cfamily.llvm-cov.reportPath = path to a llvm-cov report
- sonar.cfamily.vscoveragexml.reportsPath = path to report
- sonar.cfamily.bullseye.reportPath= path to a Bullseye report

Swift:

- sonar.swift.coverage.reportPath = Path to the report generated by llvm-covshow

Java/Kotlin:

- sonar.coverage.jacoco.xmlReportPaths = path to the jacoco test report
- sonar.junit.reportPaths = Import tests execution reports (Surefire XML format). (this is for java only)

Importing Third-Party Issues:

SonarQube won’t generate report for Third-party tools, it only imports its pre-generated report

External Analysis:

Android Lint issues	sonar.androidLint.reportPaths =/build/reports/lint-results.xml
Swift Lint issues	sonar.swift.swiftLint.reportPaths = path to generated swift lint issues
Kotlin Detekt Plugin Issues	sonar.kotlin.detekt.reportPaths = /build/reports/detect-results.xml. and more..etc

Third-Party Plugins Support:

SonarQube has extensive 3rd Party Plugins Support

Below are some of the best Free plugins that may suits for our need

HP-Fortify : https://marketplace.microfocus.com/fortify/content/fortify-sonarqube-plugin
Black Duck : https://synopsys.atlassian.net/wiki/spaces/INTDOCS/pages/622990/Black+Duck+SonarQube
Detekt Kotlin : https://github.com/detekt/sonar-kotlin
Mutation Analysis : https://github.com/devcon5io/mutation-analysis-plugin

More Info: https://docs.sonarqube.org/latest/instance-administration/plugin-version-matrix/

Limitations / Improvements

- No Built-In Support for Compiler warnings
  - This Can be achieved with a trick:
    - Compiler Phase is over, get all compiler warnings
    - Follow instructions here to create Custom issues on the Fly
    - Generic Issue Import Format | SonarQube Docs
- Currently, Pull Request quality result can not be attached to Github if PullRequest contains > 100 new commits
- Less In-built Rules for modern languages like Go/Dart/Swift/Kotlin
  - However, Allows to integrate Popular 3rd Party Approved Plugins via Marketplace from SonarQube Dashboard
- Sonar-Lint Plugin IDE plugin is only available for Eclipse, Intellij, Android Studio ,Visual Studio, VS Code. Not Available for XCode.
- Less Support to find **Fanout** issues in major languages
  - However, Good support for **Java,Kotlin** with Check style & Detekt Plugins respectively.
- No Support to detect Dead Code
- Dedicated Support is costly.
  - The support needed is very minimal. Sonar Community Forum Support is fair enough to deal to with maximum issues

THANK YOU