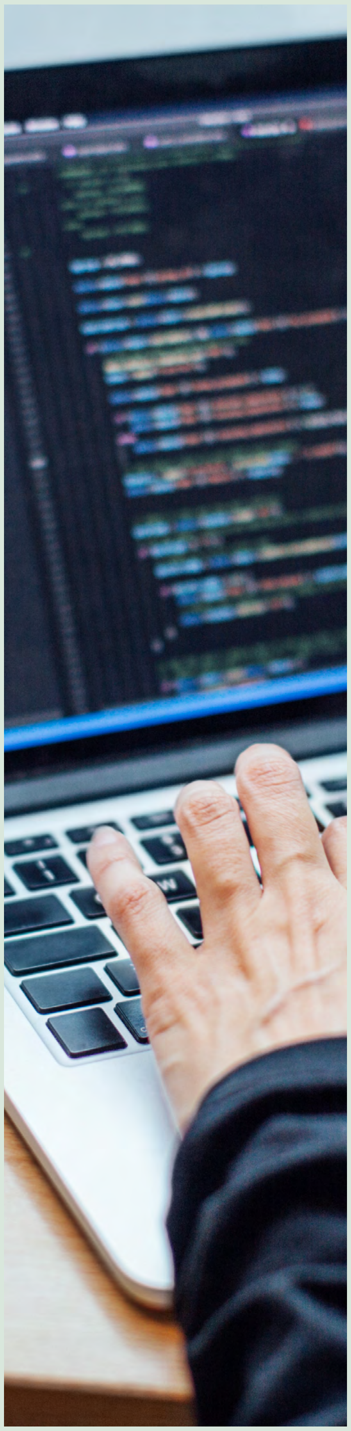
WHITE PAPER



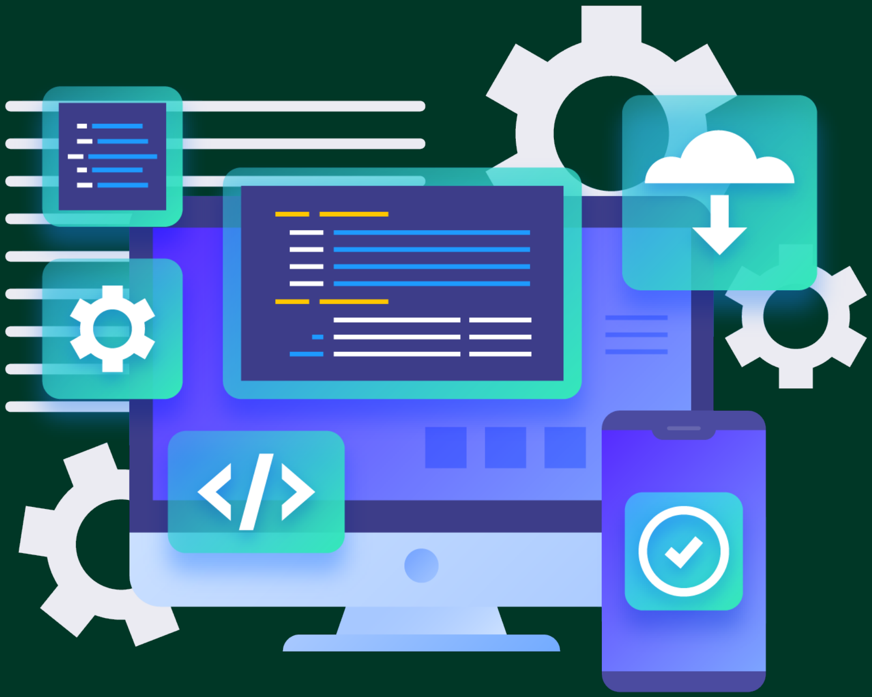
CODE QUALITY AFFAIRS: SAILING VERSUS WADING

Abstract

Code quality is an important but underrated success driver of business operations. Developers struggle with many challenges such as bugs and violations. This paper outlines how enterprises can ensure high code quality using a five-step framework. It describes a framework used by Infosys in a specific implementation that helped the client move from using code with high technical debt to keeping code clean so they can launch new features with agility.

Infosys®
Navigate your next

Code Quality Affairs: Sailing versus Wading



Introduction

The bedrock of business for most organizations today is code. Clean code is one of the most important elements of an organization's success. Yet, senior leadership often finds it challenging to keep tracking code quality. They need a comprehensive and intelligent approach that maintains code quality with least dependencies.

Need for a Planned Approach to Code Quality

While many organizations may feel that their team of experienced developers is enough to ensure code quality, the reality is different. In one example, Infosys worked with a client that had an experienced developer team, strong peer review processes, and established rules and guidelines. Yet, their code quality lagged by 40% compared to the industry benchmark. Nearly 15 bugs per 1000 lines of code would make their way to the end customer. Moreover, the client had 55,869 violations within production across 2 million lines of code.

These numbers are fairly common for most industries. Thus, ensuring high-quality code requires a systematic approach that prevents bad code from entering production environments while making code quality assurance an enterprise-wide initiative governed by best-practices.

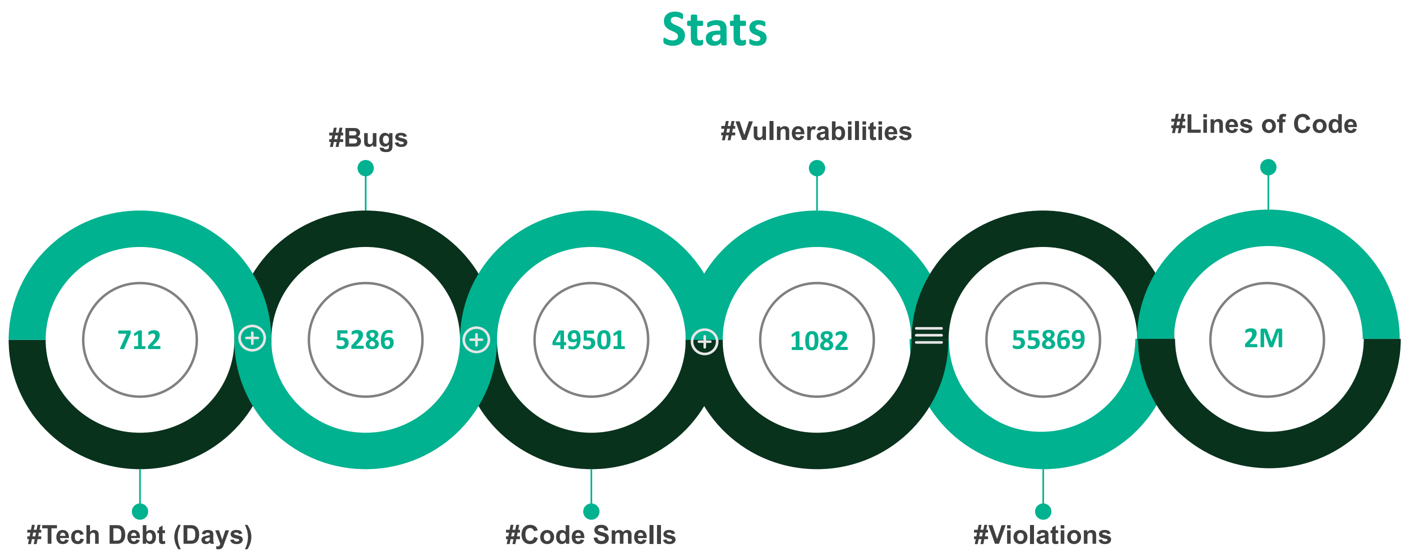


Figure 1 – The number of code defects identified by Infosys for one of its clients

Practical Steps to a Successful Code Quality Rollout

A smart approach towards ensuring code quality during every rollout is one that combines best practices, industry benchmarks, and tools. Using the Infosys case as an example, here is five-step framework that can help enterprises embark on smoother code quality journeys:

Step 1	Step 2	Step 3	Step 4	Step 5
Intention	Plan	Execute	Feedback	Package

Step 1: Intent

It is important to define the goal and identify the existing bottlenecks. In this case, despite having code review checklists and strong peer review processes, the client faced challenges such as:

- Different coding standards across teams, leading to inconsistencies and conflicts
- Manual processes, leading to errors
- No central location to track code quality and provide analytics

Thus, the goal (or intent) was clear, i.e., to automate the code review process and embed a code quality tool into the CI/CD pipelines. This would save reviewer effort by 60-70%. The tool designated for the purpose was SonarQube.

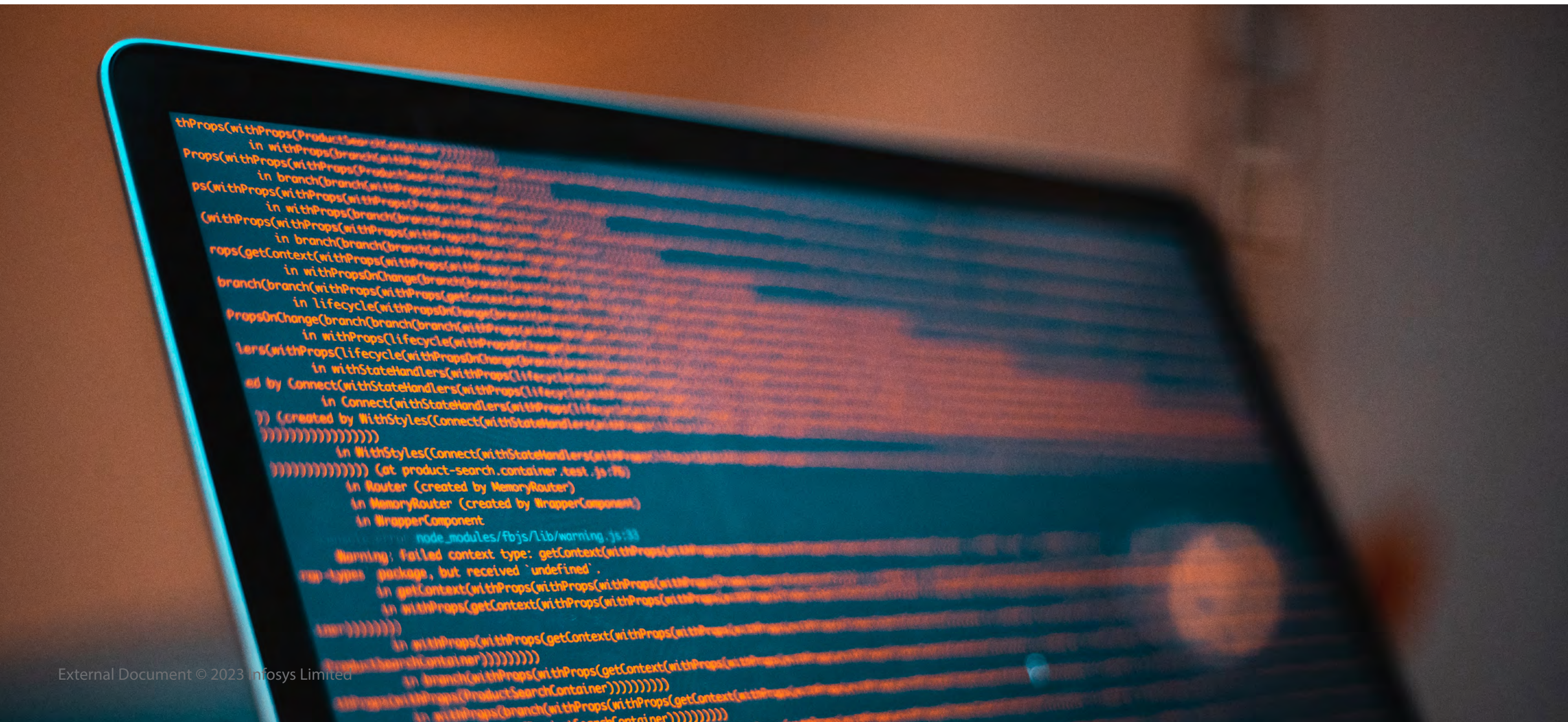
Step 2: Plan

During the planning stage, enterprises should define the number of target applications and assess integration scope. For instance, in this case, Infosys had a target to cover over 50 applications and integrate a code quality tool with more than 400 CI/CD pipelines on Azure.

Some of the key activities in this stage are:

- Streamline licensing by first gauging the number of lines of code (LoC) under scan and setting a configurable LoC threshold
- Define metrics on what is to be measured and how within the chosen tool. In this example, Infosys set up of Quality Gate
- Conduct a PoC with popular technology stacks like .NET, NodeJS, or React
- Create a timeline view. In this case, Infosys created a view covering all 50+ applications over a 4-month duration with a 4-member team
- Create the organization change management (OCM) plan. In this example, the OCM plan included emails to the leadership before starting the program, training for leaders and teams, an adoption plan pack to understand the details of the program, and a final success story email
- Articulate the results clearly. In this example, Infosys was able to turn the Quality Gate ON for all applications, thereby ensuring reliability, maintainability, code duplication, and code coverage on all unit tests

An important highlight of this stage during the Infosys implementation was how feature teams were assured that their current sprint cycles would not be disrupted when integrating the tool. Once the dashboard was ready with the details of vulnerabilities, the teams were able to connect effectively.



INNOVATION FUNNEL/METHODOLOGY

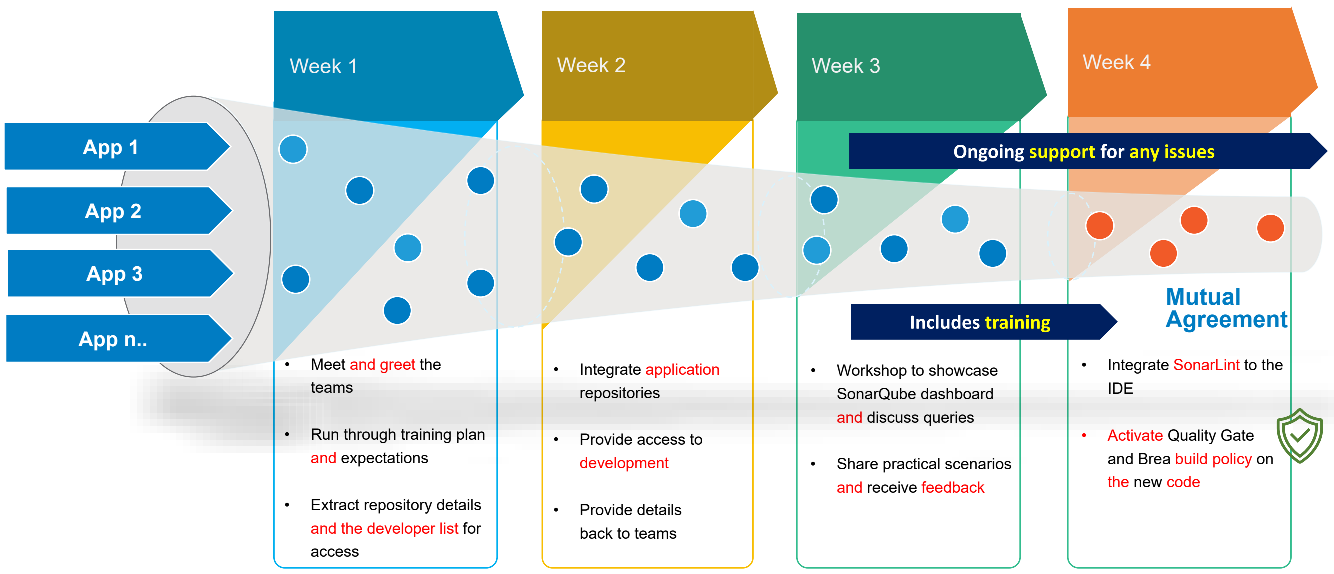

App 1
App 2
App 3
App n..
Week 1
Week 2
Week 3
Week 4
Ongoing support for any issues
Includes training
Mutual Agreement
• Meet and greet the teams
• Run through training plan and expectations
• Extract repository details and the developer list for access
• Integrate application repositories
• Provide access to development
• Provide details back to teams
• Workshop to showcase SonarQube dashboard and discuss queries
• Share practical scenarios and receive feedback
• Integrate SonarLint to the IDE
• Activate Quality Gate and Brea build policy on the new code

Figure 2 – A reference engagement model to be used with different teams

Step 3: Execute

Here are some best practices to be followed during the execution stage, as described in the context of the Infosys example:

Workload management and bonding – Infosys prepared a simple and effective template on Confluence that was updated and discussed daily to gauge progress, allocate responsibility, and manage workloads. For instance, any tasks that required intervention from other enterprise teams (such as environment team, infrastructure, and product owners) was the responsibility of the onshore team. This system enabled the four-member team, who were split onshore and offshore, to collaborate seamlessly with smooth handovers, despite the five-hour difference in time zones. It also allowed the team members to assist each other in case of illness or absence.

Zero trust policy – It is critical for teams to capture inputs from feature teams and consistently reconcile all data elements by cross verifying these in the ADO (Azure DevOps) repository. For this client, a similar exercise was attempted earlier whereby the vendor received a list of active pipelines and their URLs, tools for unit test coverage, and branch and merge strategy from the feature teams. Relying solely on this information proved costly as they could only complete 10% of the pipelines integrated with SonarQube after four months of effort. Many pipelines were not reported while some were stale and did not need integration.

Building automated reports can help highlight any mismatches. In the above case, Infosys began publishing discrepancy reports relentlessly to the feature teams and to the leadership team to sync the data. The outcome was high data quality and spot-on coverage that presented complete views to the leadership.

Pragmatism over idealism – In this Infosys engagement, the team was firm on Quality Gate and enforcing best practices. Nevertheless, there were circumstances that demanded smart thinking and exceptions. For example, a few teams running on TFS did not have a branching strategy; everything resided on the Master branch. Thus, Infosys had to relax its high standards in some areas such as introducing Quality Gating only on 'new code' since there was a varying degree of maturity across the landscape and applications carrying significant technical debt that could not be removed overnight.

Combination of skills – The diversity in the landscape demands poly-skilled individuals that are proficient in tools, technology, processes, and policies. In the above case, the Infosys team demonstrated the right skills in ADO, containerization, and branch-merge policy. They demonstrated an in-depth understanding of SonarQube and its vulnerabilities. They also showcased the right attitude to become quick learners when introducing additional tools, integrating with IDEs, building custom scripts for automation, etc. Strong communication skills aided effective training to client teams.

Step 4: Feedback

The job of fixing a broken build agent lies largely with the feature team. However, the DevOps team sometimes intervenes to make it run to save time. Such issues happen when the organization upgrades to another version or framework of .NET, leaving the build agent incompatible. It is interesting to note that SonarQube treats .NET and non-.NET scans differently. For a .NET scan, the tasks should be run in a specific order or else certain files may get exempted from the SonarQube scan. For a non-.NET code base like Node or React, there is no specific order of tasks around build and scan. The correct order for a .NET scan is:

- Prepare the analysis
- Run the build task
- Run the code analysis
- Publish the Quality Gate task

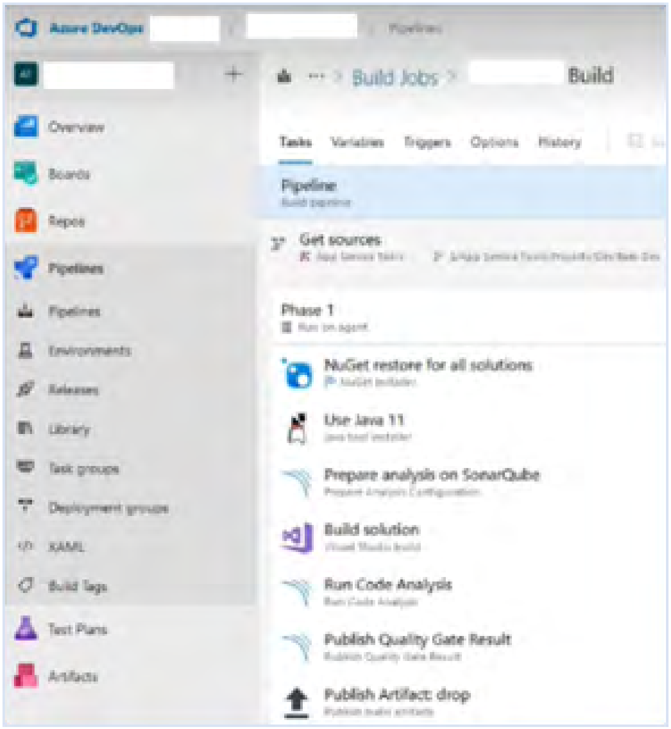


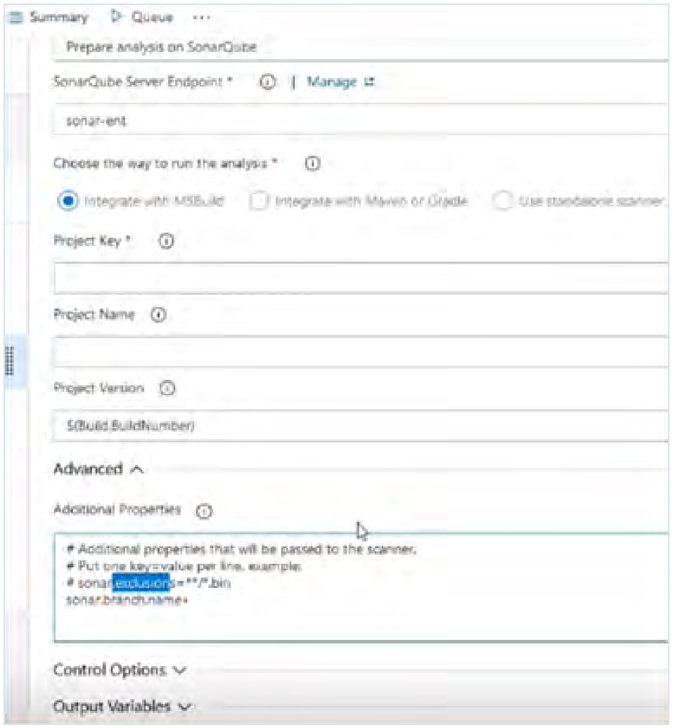
Figure 3 – Order of tasks for a .NET scan

Here are a few obstacles to consider during the execution stage:

Missing code coverage – It is important to note that SonarQube does not generate the coverage report itself. In the Infosys example, the team set up a third-party tool 'dotCover' and 'OpenCover' to produce the report as part of the build process. However, despite integrating such tools, no coverage was reported on SonarQube due to incorrect paths. For example, the path followed for dotCover is:

d:sonar.cs.dotcover.reportsPaths=dotCover.Output.html

Auto-generated code – There are cases when there will be auto-generated code by Visual Studio or a third-party tool or source code from libraries. Feature teams may want to exclude certain files from scanning as they add to unnecessary technical debt. There may also be situations where the team may want to forcefully include certain files for code quality that SonarQube may exclude by default. Infosys recommends using 'inclusions' and exclusions' as shown below.



Prepare analysis on SonarQube

SonarQube Server Endpoint *

sonar-ent

Choose the way to run the analysis *

[✓ Integrate with MSBuild] Integrate with Maven or Gradle Use standalone scanner

Project Key *

Project Name

Project Version

$(Build.BuildNumber)

Additional Properties

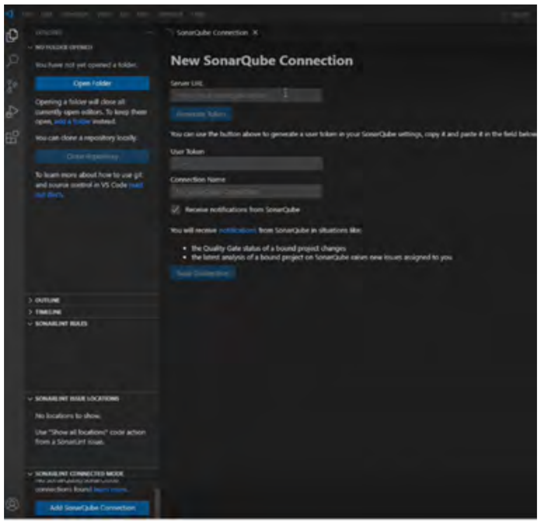
```
# Additional properties that will be passed to the scanner,
# Put one key=value per line, example:
# sonar.exclusions=**/*.bin
sonar.branch.name=
```

Figure 4 – Inclusions and exclusions

Quality shift-left with Sonarlint – Sonarlint is a free extension used with the IDEs that help quality shift-left and highlight bugs and issues at the time when developers write code. It works in two ways:

- **Connected mode** – This connects to the on-premises SonarQube server and carries any customization made around suppression of rules and addition of new rules
- **Standalone mode** – This has default ruleset and will not reflect any changes that would have been made to the ruleset on the SonarQube server

It is better to work in the 'connected' mode as there are rules that teams will need to dismiss or whitelist on the on-premises SonarQube server. This is particularly true for languages like React, Redux, and Jest (test library) where SonarQube ruleset is lacking.



NO FOLDER OPENED

You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

No locations to show.

Use "Show all locations" code action from a SonarLint issue.

connections found. learn more.

New SonarQube Connection

Server URL

You can use the button above to generate a user token in your SonarQube settings, copy it and paste it in the field below

User Token

Connection Name

[✓] Receive notifications from SonarQube

You will receive notifications from SonarQube in situations like:

- the Quality Gate status of a bound project changes
- the latest analysis of a bound project on SonarQube raises new issues assigned to you

Figure 5 – Sonarlint Integration to IDE

Step 5: Packaging

Once the implementation is successfully completed, it is important to showcase its outcomes in a way that can be easily understood by different personas. Infosys recommends using a simple dashboard underpinned by three principles:

- **Actionable insights** – The best-fit metrics are like the fertile soil where seeds of interventions can grow
- **Pull-based metrics** – A custom fully-automated solution that pulls data from discrete data sources without disturbing feature teams that are engrossed in functional delivery
- **Lightweight infrastructure** – Sufficient fidelity along with cost-effectiveness and reduced maintenance overhead

The success of metrics-driven interventions depend on capturing the attention of the leadership and teams, introducing a degree of novelty, and clearly indicating how to do things better or differently.

Code Quality Initiatives

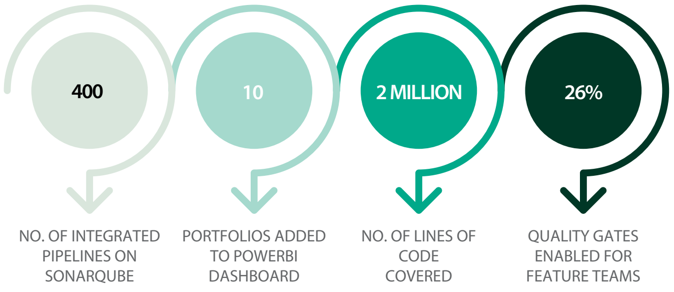


Figure 6 – Achievement outcomes or metrics of a code quality rollout

Conclusion

Organizational changes rarely follow a straight-line path. It is usually a series of advances and retreats. Achieving high quality code is possible for organizations that adopt a best practice-based approach. Infosys recommends a five-step framework, based on best practices gleaned from successful implementations. Progressing through the five steps of setting goals, planning the project, executing the activities, capturing feedback, and showcasing the outcomes can help organizations maintain a high level of code quality.

About the author:

Adarsh Mehrotra, Senior Industry Principal Consultant – Infosys

Adarsh has 18 years of IT experience and leads the DevOps & SRE practice at the Quality Consulting unit (EQS). He has led end to end complex large-scale DevOps, Engineering & Agile transformations in the last six-seven years in UK, Australia & Singapore. He has been instrumental in setting up DevOps CoE for greenfield digital set-ups in telcos as well as banks. He is a regular speaker at a number of covered forums on DevOps & SRE across the globe. He spearheads the EQS practice with 175+ consultants across the globe who are involved in cutting-edge technology changes with 40+ clients.