CODE QUALITY

WHAT IS CODE QUALITY

The code quality is important, as it impacts the overall software quality. And quality impacts how safe, secure, and reliable your codebase is.

High quality is critical for many development teams today. And it's especially important for those developing safety-critical systems.

CODE QUALITY ANALYSIS: GOOD CODE VS BAD CODE

Good code is high quality. And it’s clean code. It stands the test of time. Bad code is low quality. It won’t last long.

Essentially, code that is considered good:

- Does what it should.
- Follows a consistent style.
- It is easy to understand.
- Has been well-documented.
- It can be tested.

TESTING ISN’T ENOUGH

Programmers aren’t perfect. Manual code reviews and testing will never find every error in the code.

A study on “Software Defect Origins and Removal Methods” found that individual programmers are less than 50% efficient at finding bugs in their own software. And most forms of testing are only 35% efficient. This makes it difficult to determine quality.

CODING ERRORS LEAD TO RISK

The quality of code in programming is important. When code is low quality, it might introduce safety or security risks. If the software fails — due to a security violation or safety flaw — the results can be catastrophic or fatal.

QUALITY IS EVERYONE’S RESPONSIBILITY

Quality is everyone’s job. The developer. The tester. The manager. High quality should be the goal throughout the development process.

HOW TO MEASURE CODE QUALITY? CODE QUALITY METRICS

There’s no one way to measure the quality of your code. What you measure may be different from what other development team measures.

KEY CODE QUALITY ASPECTS TO MEASURE

Reliability

Reliability measures the probability that a system will run without failure over a specific period of operation. It relates to the number of defects and availability of the software.

Number of defects can be measured by running a static analysis tool. Software availability can be measured using the mean time between failures (MTBF). Low defect counts are especially important for developing a reliable codebase.

Maintainability

Maintainability measures how easily software can be maintained. It relates to the size, consistency, structure, and complexity of the codebase. And ensuring maintainable source code relies on a number of factors, such as testability and understandability.

You can’t use a single metric to ensure maintainability. Some metrics you may consider to improve maintainability are the number of stylistic warnings and Halstead complexity measures. Both automation and human reviewers are essential for developing maintainable codebases.

Testability

Testability measures how well the software supports testing efforts. It relies on how well you can control, observe, isolate, and automate testing, among other factors.

Testability can be measured based on how many test cases you need to find potential faults in the system. Size and complexity of the software can impact testability. So, applying methods at the code level — such as cyclomatic complexity — can help you improve the testability of the component.

Portability

Portability measures how usable the same software is in different environments. It relates to platform independency.

There isn’t a specific measure of portability. But there are several ways you can ensure portable code. It’s important to regularly test code on different platforms, rather than waiting until the end of development. It’s also a good idea to set your compiler warning levels as high as possible — and use at least two compilers. Enforcing a coding standard also helps with portability.

Reusability

Reusability measures whether existing assets — such as code — can be used again. Assets are more easily reused if they have characteristics such as modularity or loose coupling.

Reusability can be measured by the number of interdependencies. Running a static analyzer can help you identify these interdependencies.

WHICH CODE QUALITY METRICS TO USE

There are several metrics you can use to quantify the quality of your code.

Defect Metrics

The number of defects — and severity of those defects — are important metrics of overall quality.

This includes:

- Identification of the stage in which the defect originates.
- Number of open defect reports.
- Time to identify and correct defects.
- Defect density (e.g., number of defects per lines of code).

COMPLEXITY MATRICS

Complexity metrics can help in measuring quality.

Cyclomatic complexity measures of the number of linearly independent paths through a program’s source code.

Another way to understand quality is by calculating Halstead complexity measure

These measure:

- Program vocabulary
- Program length
- Calculated program length
- Volume
- Difficulty
- Effort

HOW TO IMPROVE CODE QUALITY AND CODE QUALITY ANALYSIS

Measuring quality helps you understand where you’re at. After you’ve measured, you can take steps to improve overall quality.

Here are four ways you can improve the quality of your code:

1. Use a coding standard.
2. Analyze code — before code reviews.
3. Follow code review best practices.
4. Refactor legacy code (when necessary)

Using a coding standard is one of the best ways to ensure high quality code.

A coding standard makes sure everyone uses the right style. It improves consistency and readability of the codebase. This is key for lower complexity and higher quality
How to Do It

The best way to use a coding standard is to:

- Train your developers
- Help them to comply with it