What is code coverage?

Internal reference: topics/06-1.md

Introduction

Code coverage is a software testing metric that determines the number of lines of code that is successfully validated under a test procedure. This metric helps to analyze how comprehensively a software is verified. Developing enterprise-grade software products is the ultimate goal of any software company. However, to accomplish this goal, companies have to ensure that the software they develop meets all the essential quality characteristics, i.e. according to ISO 9126 standard that is – `functionality, usability, reliability, efficiency, security, portability and maintainability`. This can only be possible by thoroughly reviewing the software product. Along with handing off the software to the QA engineers for bug tracking, it is imperative to analyze, monitor, measure test activities. This means, software testing metrics to evaluate the test suite's effectiveness and completeness should be considered. `Code coverage` is one such software testing metric that can help in assessing the test performance and quality aspects of any software. Such an insight will equally be beneficial to the development and QA team. For developers, this metric can help in dead code detection and elimination. On the other hand, for QA, it can help to check missed or uncovered test cases. They can track the health status and quality of the source code while paying more heed to the uncaptured parts of the code.

Benefits of Code Coverage

Before we list down the benefits, let's first burst a few myths. Code coverage analysis can only be used for the validation of test cases that are run on the source code and not for the evaluation of the software product. Also, it neither evaluates whether the source code is bug-free nor proves if a written code is correct. Then, why is it important you ask? Here's why you should care about this analysis:

1. `Easy maintenance of code base` – Writing scalable code is crucial to extend the software program through the introduction of new or modified functionalities. However, it is difficult to determine whether the written code is scalable. It can prove to be a useful metric in that context. The analysis report will help developers to ensure code quality is well-maintained and new features can be added with little-to-no efforts.
2. `Exposure of bad code` – Continuous analysis will help developers to understand bad, dead, and unused code. As a result, they can improve code-writing practices, which in turn, will result in better maintainability of the product quality.
3. `Faster time to market` – With the help of this metric, developers can finish the software development process faster, thereby increasing their productivity and efficiency. As a result, they will be able to deliver more products, allowing companies to launch more software applications on the market in lesser time. This will undoubtedly lead to increased customer satisfaction and high ROI.

How is it measured?

To calculate the code coverage percentage, simply use the following formula:

```
Code Coverage Percentage =
(
    Number of lines of code executed by a testing algorithm
    divided by
    Total number of lines of code in a system component
)
multiplied by 100.
```

For example: If the software you are testing contains a total of 100 lines of code and the number of lines of code that is actually validated in the same software is 50, then the code coverage percentage of this software will be 50 percent.

Looking at the example above, you might crave to achieve 100 percent coverage for your software product. You may think, the more the coverage, the better the code quality of any software program. However, this isn’t true. So, what ideal coverage percentage developers and testers should aim for?

What is an ideal code coverage percent?

Striking 100 percent code coverage means the code is 100 percent bugless. No error indicates that test cases have covered every criterion and requirement of the software application. So, if that’s the case, how do we evaluate if the test scripts have met a wide range of possibilities? What if the test cases have covered the incorrect requirements? What if test cases have missed on some important requirements? So, that drills down to the fact that, if a good software product built on 100 percent irrelevant test case coverage, then the software will undoubtedly compromise on quality.

The only focus and goal of developers and testers should be to write test scripts that aren’t vague. Don’t focus to achieve 100 percent coverage. The analysis should be clubbed with scalable, robust test scripts, covering every functional and non-functional area of the source code.

Code coverage criteria

To measure the lines of code that are actually exercised by test runs, various criteria are taken into consideration. We have outlined below a few critical coverage criteria that companies use.

1. `Function Coverage` – The functions in the source code that are called and executed at least once.
2. `Statement Coverage` – The number of statements that have been successfully validated in the source code.
3. `Branch or Decision Coverage` – The decision control structures (loops, for example) that have executed fine.

4. `Condition Coverage` – The Boolean expressions that are validated and that executes both TRUE and FALSE as per the test runs.
5. `Path Coverage` – The flows containing a sequence of controls and conditions that have worked well at least once.

Summing Up

In this fast-paced technology-driven world, developers and testers have to intensify the rapidity of their software development life-cycles. And to handle tight deadlines, software engineers must build only good code. Hence, good code quality is what every developer or tester is aiming for. With a code coverage analysis report, they can track the proportion of code that worked well under test scenarios. This insight will act like a feedback report, which will help developers to write good and clean source code. This ultimately will result in improved code quality, positively impacting the software quality.

However, depending on coverage metrics solely for assessing code health isn't a good option. Code coverage analysis and code reviews, along with your QA efforts, can be one powerful way of improving the functionality of code.
