← Back to Blog

What Is Branch Coverage and What Does It Really Tell You?

What is branch coverage? That's the question we'll answer today. You'll learn what this metric is all about and how it can help you.

Published February 18, 2021 • Carlos Schults

Newcomers to automated testing—and unit testing in particular—often get confused when it comes to assessing the quality of their test cases. How can you know that the tests you write really verify your app is working correctly? Is it possible to know for sure whether the tests you have are enough for your application?

Let’s get started.

Table of Contents

Branch Coverage: A Quick Definition

Branch coverage is an important metric in that it can help a team or organization assess whether an application has been tested to completion.

As you'll soon see, branch coverage is more nuanced than other metrics. A different metric can be at 100%, while branch coverage is lower. By only tracking the other metric, a team can have an unjustified degree of confidence in their code, and important defects might go unnoticed until they manifest in production.

Branch Coverage: How It Differs From Similar Metrics

So, in short, we can say that branch coverage is a subset of code coverage. It's a more specialized version of code coverage that focuses on a specific aspect—i.e., ensuring that every branch or path is tested.

Branch vs. Statement Coverage

Here's where things can get somewhat confusing. Some people struggle to understand the difference between these two metrics. After our explanation and example, you'll hopefully easily understand how they differ.

Let's start with statement coverage. This metric simply tells you the ratio of statements in an application that are currently under testing. Branch coverage, as we've seen, is about whether all branches—or paths of execution—in an application are under test.

Branch vs. Line Coverage

Finally, let's compare branch coverage and line coverage. We'll define line coverage and then it'll hopefully be clear how they differ.

Line coverage is closer to statement coverage. It indicates the ratio of lines exercised by tests. The difference between line coverage and statement coverage is that the correspondence between statements and lines isn't always one to one. Depending on the programming language, a statement can span multiple lines and a single line could contain multiple statements.

So, while 100% statement coverage necessarily implies 100% line coverage, the opposite isn't true. A line can contain multiple statements, but it's possible not all of them will be executed.

Finally, branch coverage differs from line coverage in a similar way to which it differs from

But, in our context, cyclomatic complexity is essential. It helps to determine the minimum number of test cases you need to comprehensively test a given piece of code. So striving to keep cyclomatic complexity low is a good goal to have if you want to accomplish higher levels of branch coverage.

The Good and the Bad

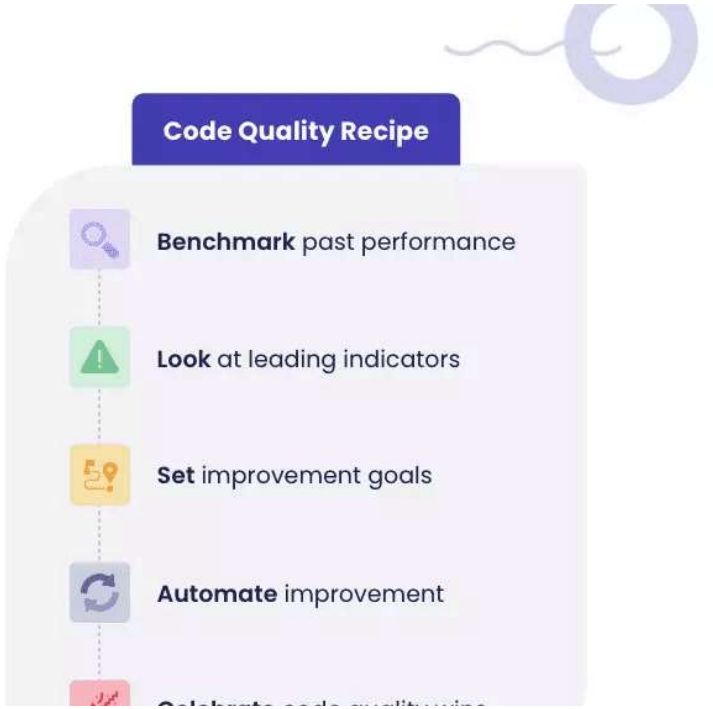
This is no silver bullet in software development. It's important for you to understand that, no matter how good a metric is, it doesn't tell you everything. Also, Goodhart's Law—or, more specifically, the generalization of it made by Marilyn Strathern—warns you that any metric that becomes a target loses its value as a metric. So we shouldn't forget that metrics can be gamed.

This isn't to say there aren't good things to be said about branch coverage. Here are some:

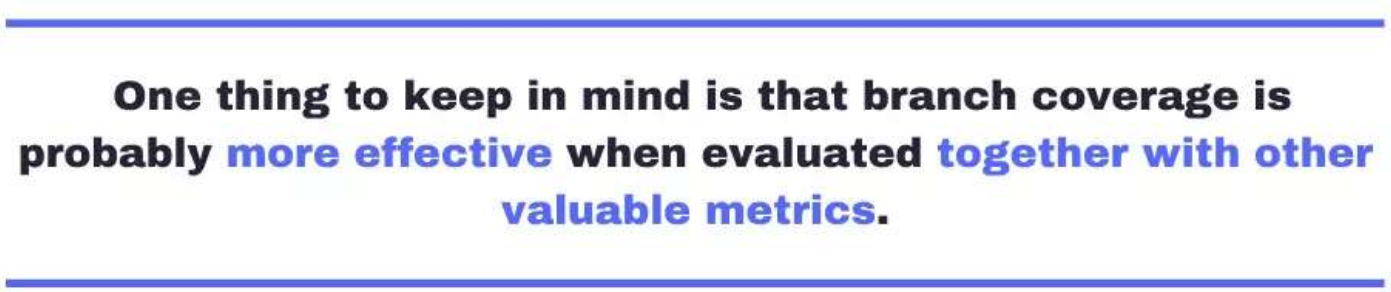
Improve
Code Quality

Teams who improve code quality with LinearB see:

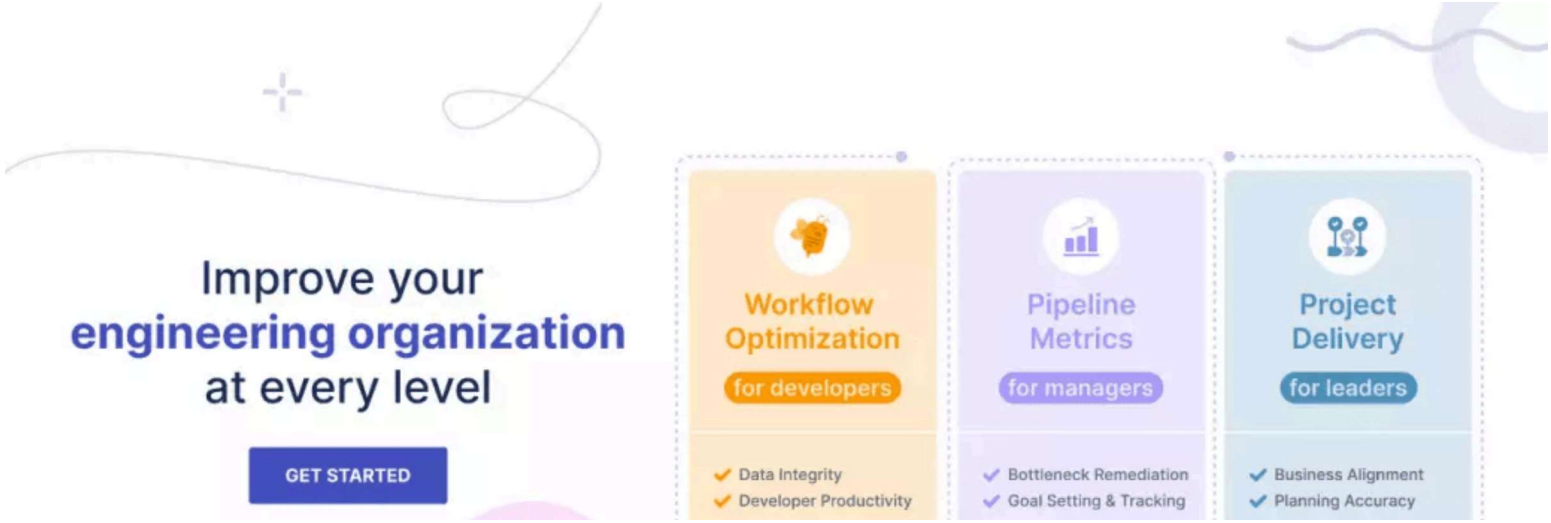
- **Faster MTTR**
- **Less code churn**
- **Shorter Cycle Time**
- **Higher performing apps**



application against defects. Even though it isn't perfect—like any other metric—branch coverage is an important way to help teams that need an objective method to assess the health of their test suites.



One thing to keep in mind is that branch coverage is probably more effective when evaluated together with other valuable metrics. There are other indicators that can help predict the quality of codebases in general—such as rework or code churn, for instance.



Further Reading

LinearB

MARCH 14, 2024 • NATALIE BREUER

How Rabbit Care Used LinearB to Build a Culture of Transparency While Scaling 10x

Read Now →

LinearB

MARCH 11, 2024 • NECCO CERESANI

LinearB Best Practices for Integrating Jira

Read Now →

MARCH 7, 2024 • LINEARB

Software Engineering Intelligence 101 - Everything You Need to Know About SEIPs

Read Now →