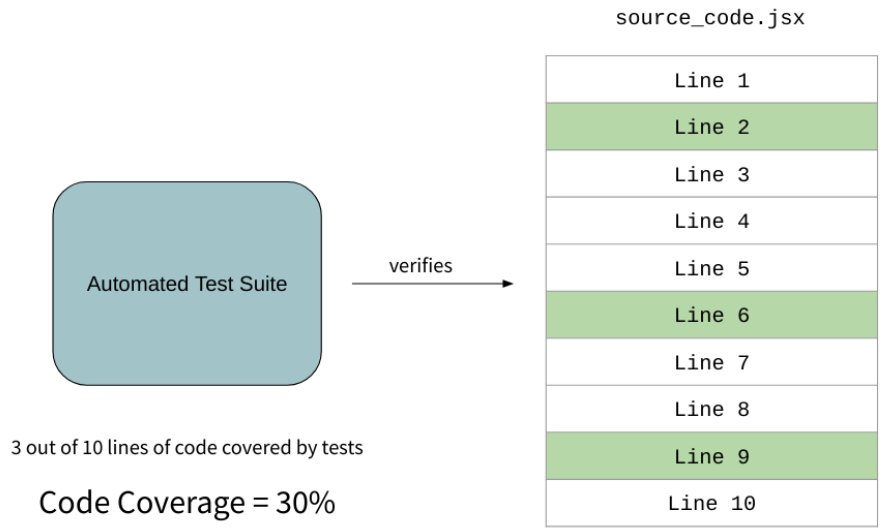
13.CODE COVERAGE

Fact Card - Code Coverage	
Category	Quality
Typical usage pattern	Set a goal for minimum code coverage
Impact	Developers write useless tests which do not verify anything
Suggested to use with metrics	None
Suggested to use instead of metrics	None
Metric recommended?	No

Code coverage is one of the most popular metrics that the engineering teams use to measure and showcase their quality. This chapter will analyze whether code coverage is a good metric for quality measurement.

What is Code Coverage?

Code coverage refers to the percentage of lines in source code that is covered when the automated tests execute.



Let us take an example to explain this. Let us assume that we have a file with 10 lines in the source code. We also have an automated test suite that we run before releasing the code to the customer.

Let us assume that these automated tests when they run, execute 3 out of the 10 lines of code. This would mean that only 30% of the code is tested by the test suite. This 30% is called the line coverage or code coverage.

There are other ways in which the code coverage can be measured, in addition to the number of lines of code. Some of them are:

1. Branch/Decision coverage
2. Function/Method coverage

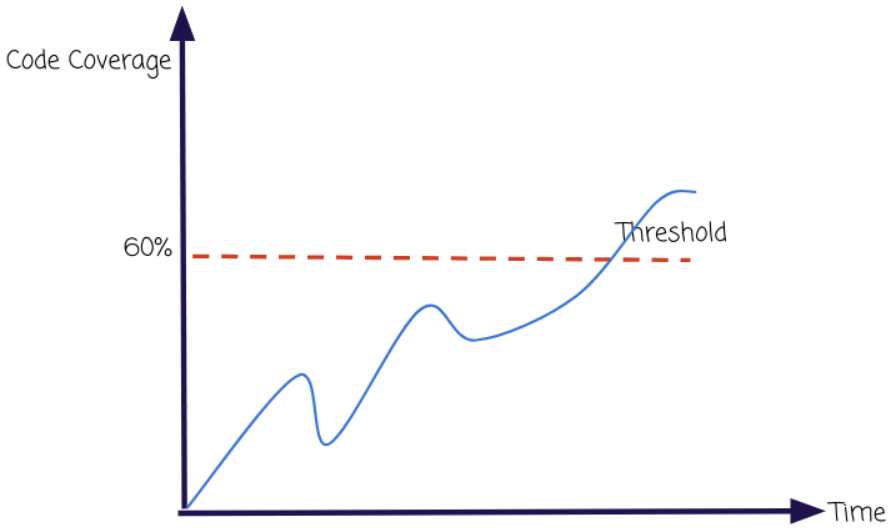
No matter which kind of coverage you use, the underlying concept remains the same.

Code Coverage is generally measured using tools like Sonarqube, Jacoco, Cobertura, etc.

How do teams generally use this metric?

The teams use the code coverage in any of the following ways.

1. Tracking this metric over time and creating a trend chart
2. Setting a bare minimum threshold of code coverage to pass a build
3. Setting a goal of reaching X% code coverage by a specific timeframe



Why is this a bad idea?

So far, everything looks logical and common sense. However, treating just the code coverage as a quality metric is a bad idea. Here is why.

1. Code coverage tells us only whether people are writing tests or not. It does not tell whether the tests are good or not. This means that code coverage is only used to get into the practice of writing tests, not anything else.
2. It is very easy to write useless tests that verify nothing but cover all the code. When the management treats code coverage as a golden metric and imposes a threshold on the team, the developers find a way to quickly write useless tests just to meet the threshold. The illustration at the end of this chapter explains this in detail.
3. This will also add to increased build and execution times for no good reason since these useless tests will run in every single build. This will delay every potential