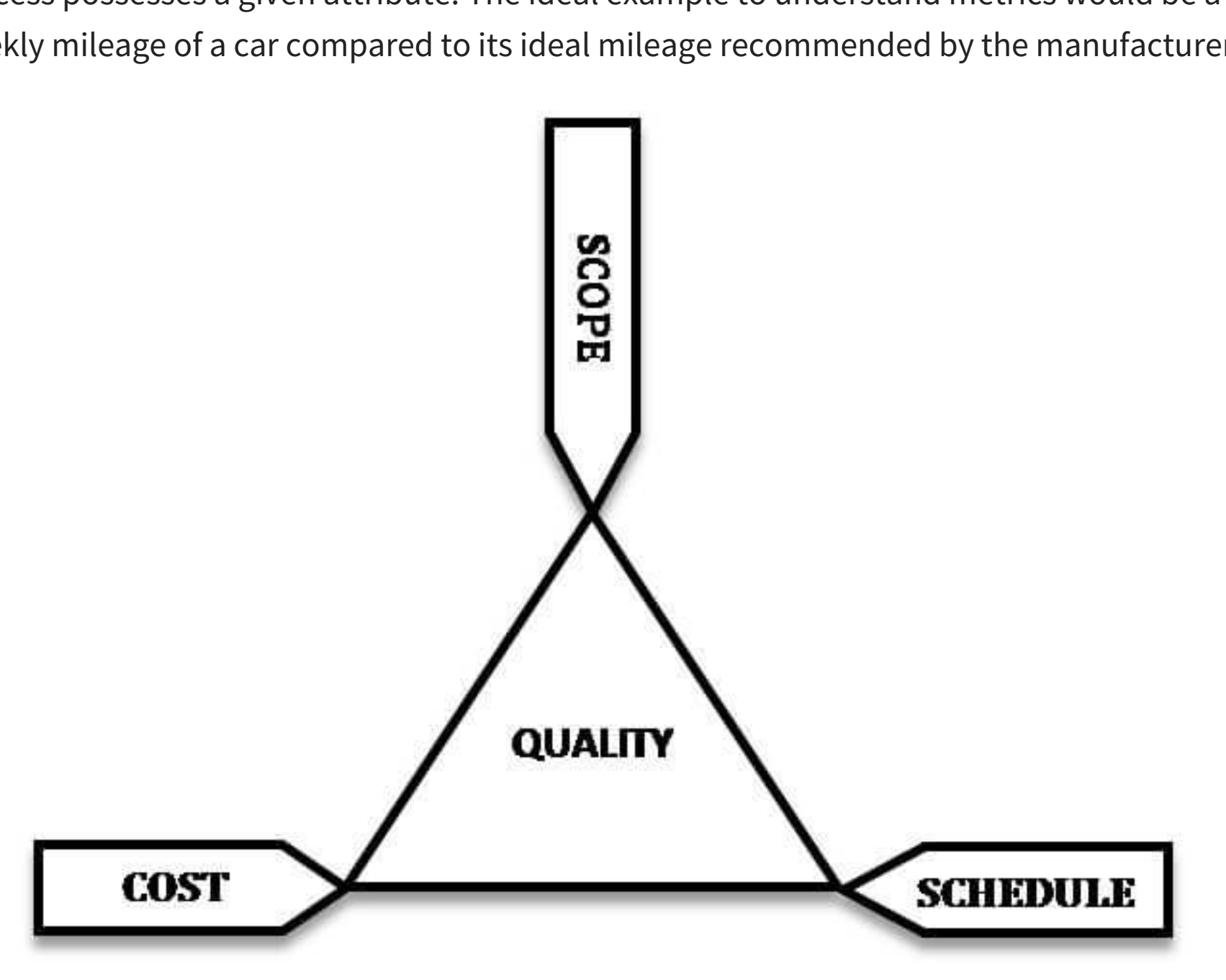
Software Testing Metrics: What is, Types & Example

Software Testing Metrics

Software Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.



Software testing metrics - Improves the efficiency and effectiveness of a software testing process.

Software testing metrics or software test measurement is the quantitative indication of extent, capacity, dimension, amount or size of some attribute of a process or product.

Example for software test measurement: Total number of defects

In this tutorial, you will learn-

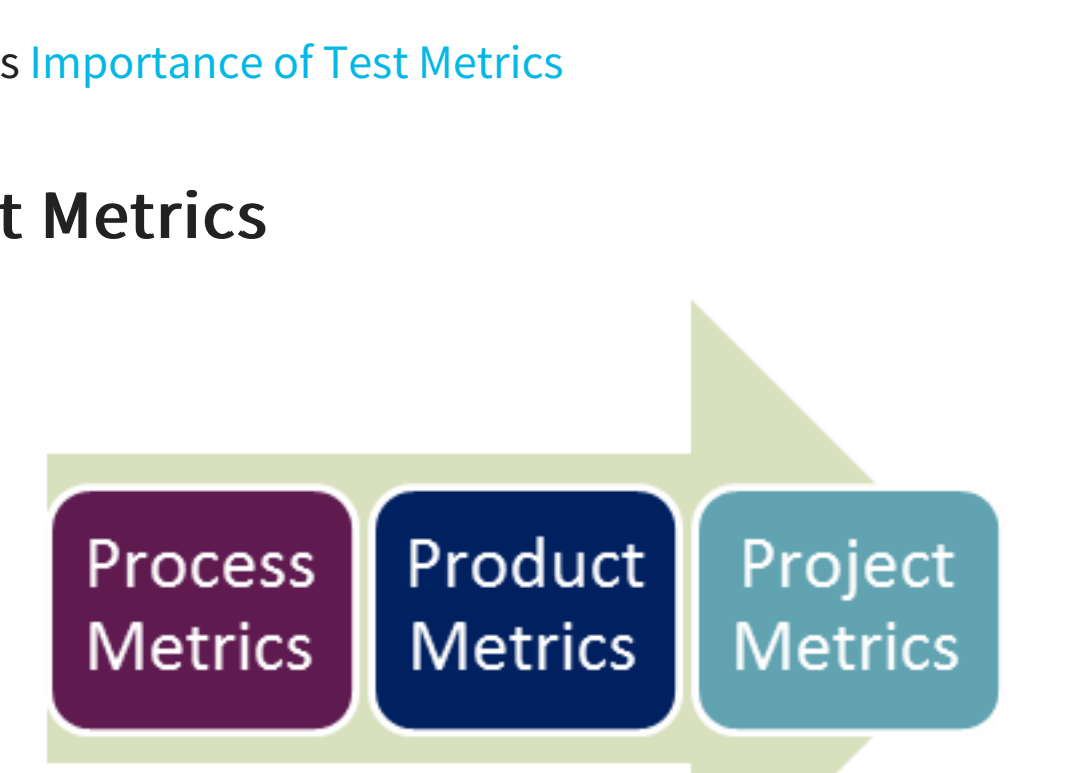
Why Test Metrics are Important?

```
"We cannot improve what we cannot measure" and Test Metrics helps us
to do exactly the same.
```

- Take decision for next phase of activities
- Evidence of the claim or prediction
- Understand the type of improvement required
- Take decision or process or technology change

Read more about its Importance of Test Metrics

Types of Test Metrics



- **Process Metrics:** It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle)
- **Product Metrics:** It deals with the quality of the software product
- **Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

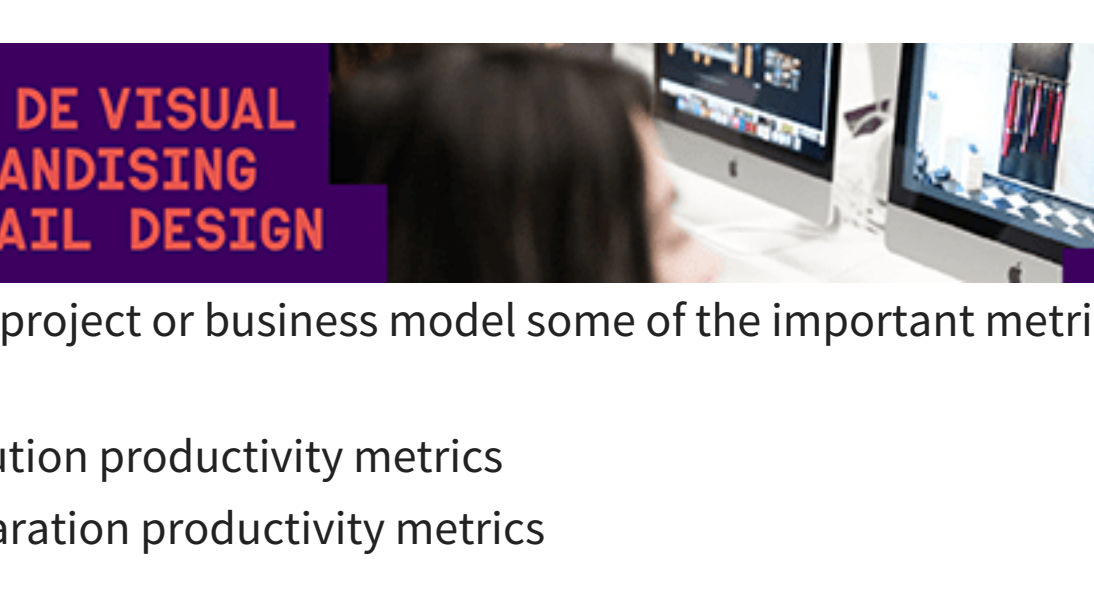
Identification of correct testing metrics is very important. Few things need to be considered before identifying the test metrics

- Fix the target audience for the metric preparation
- Define the goal for metrics
- Introduce all the relevant metrics based on project needs
- Analyze the cost benefits aspect of each metrics and the project lifestyle phase in which it results in the maximum output

Manual Test Metrics

In Software Engineering, Manual test metrics are classified into two classes

- **Base Metrics**
- **Calculated Metrics**



Base metrics is the raw data collected by Test Analyst during the test case development and execution (**# of test cases executed, # of test cases**). While calculated metrics are derived from the data collected in base metrics. Calculated metrics is usually followed by the test manager for test reporting purpose (**% Complete, % Test Coverage**).

Depending on the project or business model some of the important metrics are

- Test case execution productivity metrics
- Test case preparation productivity metrics
- Defect metrics
- Defects by priority
- Defects by severity
- Defect slippage ratio

Test Metrics Life Cycle

Different stages of Metrics life cycle	Steps during each stage
• Analysis	• Identification of the Metrics • Define the identified QA Metrics
• Communicate	• Explain the need for metric to stakeholder and testing team • Educate the testing team about the data points to need to be captured for processing the metric
• Evaluation	• Capture and verify the data • Calculating the metrics value using the data captured
• Report	• Develop the report with an effective conclusion • Distribute the report to the stakeholder and respective representative • Take feedback from stakeholder

How to calculate Test Metric

Sr#	Steps to test metrics	Example
1	Identify the key software testing processes to be measured	• Testing progress tracking process
2	In this Step, the tester uses the data as a baseline to define the metrics	• The number of test cases planned to be executed per day
3	Determination of the information to be followed, a frequency of tracking and the person responsible	• The actual test execution per day will be captured by the test manager at the end of the day
4	Effective calculation, management, and interpretation of the defined metrics	• The actual test cases executed per day
5	Identify the areas of improvement depending on the interpretation of defined metrics	• The Test Case execution falls below the goal set, we need to investigate the reason and suggest the improvement measures

Example of Test Metric

To understand how to calculate the test metrics, we will see an example of a percentage test case executed.

To obtain the execution status of the test cases in percentage, we use the formula.

```
Percentage test cases executed= (No of test cases executed/ Total no
of test cases written) X 100
```

Likewise, you can calculate for other parameters like **test cases not executed, test cases passed, test cases failed, test cases blocked, etc.**

Test Metrics Glossary

- **Rework Effort Ratio** = (Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100
- **Requirement Creep** = (Total number of requirements added/No of initial requirements)X100
- **Schedule Variance** = (Actual efforts – estimated efforts) / Estimated Efforts) X 100
- **Cost of finding a defect in testing** = (Total effort spent on testing/ defects found in testing)
- **Schedule slippage** = (Actual end date – Estimated end date) / (Planned End Date – Planned Start Date) X 100
- **Passed Test Cases Percentage** = (Number of Passed Tests/Total number of tests executed) X 100
- **Failed Test Cases Percentage** = (Number of Failed Tests/Total number of tests executed) X 100
- **Blocked Test Cases Percentage** = (Number of Blocked Tests/Total number of tests executed) X 100
- **Fixed Defects Percentage** = (Defects Fixed/Defects Reported) X 100
- **Accepted Defects Percentage** = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100
- **Defects Deferred Percentage** = (Defects deferred for future releases /Total Defects Reported) X 100
- **Critical Defects Percentage** = (Critical Defects / Total Defects Reported) X 100
- **Average time for a development team to repair defects** = (Total time taken for bugfixes/Number of bugs)
- **Number of tests run per time period** = Number of tests run/Total time
- **Test design efficiency** = Number of tests designed /Total time
- **Test review efficiency** = Number of tests reviewed /Total time
- **Bug find rote or Number of defects per test hour** = Total number of defects/Total number of test hours

Prev Report a Bug Next

Testing Tutorials