**What is Statement coverage in testing?**

Statement coverage is a white box testing technique, which involves the execution of all the statements at least once in the source code. It is a metric, which is used to calculate and measure the number of statements in the source code which have been executed. Using this technique we can check what the source code is expected to do and what it should not. It can also be used to check the quality of the code and the flow of different paths in the program. The main drawback of this technique is that we cannot test the false condition in it.

(Statement coverage = No of statements Executed/Total no of statements in the source code \* 100)

Example:

Read A  
Read B  
if A>B  
Print “A is greater than B”  
else  
Print "B is greater than A"  
endif

Set1 :If A =5, B =2  
No of statements Executed: 5  
Total no of statements in the source code: 7  
Statement coverage =5/7\*100 = 71.00 %

Set1 :If A =2, B =5  
No of statements Executed: 6  
Total no of statements in the source code: 7  
Statement coverage =6/7\*100 = 85.20 %

Tools for test planning and design

**1. Postman**

Being originally a Chrome browser plugin, Postman now extends their solution with the native version for both Mac and Windows.

[Check out here](https://www.softwaretestingmaterial.com/install-postman/)

Postman is a good choice for API testing for those who don’t want to deal with coding in an integrated development environment using the same language as the developers.

* Easy-to-use REST client
* Rich interface which makes it easy to use
* Can be used for both automated and exploratory testing
* Can be run on Mac, Windows, Linux & Chrome Apps
* Has a bunch of integrations like support for Swagger & RAML formats
* Has Run, Test, Document and Monitoring Features
* Doesn’t require learning a new language
* Enable users to easily share the knowledge with the team as they can package up all the requests and expected responses, then send to their colleagues.

Website: <https://www.getpostman.com/>

**2. SoapUI**

SoapUI is a headless functional testing tool dedicated to API testing, allowing users to test REST and SOAP APIs and Web Services easily.

Using SoapUI, users can get the full source and build the preferred features besides these abilities:

* Create test quickly and easily with Drag and drop, Point-and-click
* Quickly create custom code using Groovy
* Powerful data-driven testing: Data loaded from files, and databases, and Excel so that they can simulate how consumers interact with the APIs
* Create complex-scenarios & support asynchronous testing
* Reusability of Scripts: load tests and security scans can be reused for functional test cases in a just several steps

Website: <https://www.soapui.org/>

**Pricing:** **Free – $659/year**

**3. JMeter**

JMeter (open source) is widely used for functional API testing although it is actually created for load testing.

* Supports replaying of test results
* Automatically work with CSV files, allowing the team to quickly create unique parameter values for the API tests.
* Users can include the API tests in CI pipelines thank to the integration between JMeter and Jenkins
* It can be used for both static as well as dynamic resources performance testing

Website: <https://jmeter.apache.org/>

**Pricing: Open source**

**4. Rest-Assured**

Rest-Assured is an open-source Java Domain-specific language that makes testing REST service more simple.

* Have a bunch of baked-in functionalities, which means users don’t have to code things from scratch.
* Integrates seamlessly with Serenity automation framework, so that users can combine the UI and REST tests all in one framework that generates awesome reports.
* Support BDD Given/When/Then syntax
* Users don’t necessarily need to be an HTTP expert

Website: <http://rest-assured.io/>

1. **PractiTest:**

# Best Test Management Tools of 2019

Last Updated on August 2, 2019 by [Rajkumar](https://www.softwaretestingmaterial.com/author/smrajkumar27gmail-com/) [Leave a Comment](https://www.softwaretestingmaterial.com/test-management-tools/" \l "respond)

We spent 48 hours evaluating and researching 12 different Test Management Tools. We looked for features that should be available in an ideal test management tool. We looked at user reviews and compare products’ features to determine which tools help you in your project development. Based on our research, we believe the following tools drive your project in a better direction. Here we are not giving any ranking to any tool. Every tool in the list below has decent features but its very rare to find free test case management tools but almost all the test management tools come with free trials.

In this tutorial, we are going to see the following.

* 1. [What do we expect from a Test Management Tool](https://www.softwaretestingmaterial.com/test-management-tools/" \l "What-we-expect-from-a-Test-Management-Tool)
* 2. [Why Test Management Tools](https://www.softwaretestingmaterial.com/test-management-tools/" \l "Why-Test-Management-Tools)
* 3. [Popular Test Management Tools](https://www.softwaretestingmaterial.com/test-management-tools/" \l "Popular-Test-Management-Tools)
* 4. [Features of Test Management Tools](https://www.softwaretestingmaterial.com/test-management-tools/" \l "Features-of-Test-Management-Tools)

[](https://i2.wp.com/www.softwaretestingmaterial.com/wp-content/uploads/2018/07/Test-Management-Tools.png?ssl=1)

### ****What do we expect from a Test Management Tool?****

One who wants to use a Test Management Tool expect the following:

1. Easy installation
2. Creating and maintaining Projects
3. User and User Role creation
4. Requirements management
5. Test Plans creation
6. Test Cases creation
7. Test Cases execution
8. Reporting system
9. Defect tracking system
10. Importing and exporting system
11. Integration with other tools

### ****Why Test Management Tools?****

In the earlier section, we have seen what we expect from a Test Management Tools. So an ideal test management tool should provide features like creating projects, users, test plans, test cases, reporting system and so on. Test management tools save the time of testers in the testing process and also streamline the testing process.

### ****Popular Test Management Tools:****

Let’s see some of the popular tools for the test management process.

1. PractiTest
2. TestLodge
3. TestCaseLab
4. qTest
5. Zephyr
6. Test Collab
7. TestLink
8. Quality Center
9. Test Rail
10. Kualitee
11. Testuff
12. QMetry
13. TestFLO for JIRA
14. Qase
15. IBM Rational Quality Manager
16. Panaya
17. Bugzilla Testopia
18. XQual
19. QAComplete
20. QACoverage
21. Plutora Test
22. Inflectra
23. TestMonitor
24. Meliora Testlab
25. Borland Silk Central
26. Gemini
27. Fitnesse
28. Tarantula
29. RTH Turbo

[List of Popular API Testing Tools](https://www.softwaretestingmaterial.com/best-api-testing-tools/)

### ****Features of Test Management Tools:****

Now in this section we will see some of the above tools in detail.

**1. PractiTest:**

[](https://i2.wp.com/www.softwaretestingmaterial.com/wp-content/uploads/2018/07/PractiTest-Logo.png?ssl=1)

PractiTest is an end-to-end Test Management tool for quality assurance testing management. PractiTest offers a 30-days free trial. PractiTest integrates with some of the systems such as JIRA, Redmine, Pivotal tracker and other. PractiTest is able to work with automized tests of Selenium by using API.

**Features:**

* Create and organize tests based on test cycles, sprints
* Easy requirement management and traceability between requirements, tests, and issues
* Seamlessly integrates automation, CI, and bug tracking tools.
* Ability to report issues directly from emails
* Use advanced features to save time and money with anti-bug duplicates
* Visualize your data in the most advanced way using dashboards and reports
* Reuse tests and correlate results across different releases and products.

**2. TestLodge:**

[](https://i1.wp.com/www.softwaretestingmaterial.com/wp-content/uploads/2018/07/TestLodge-Logo.png?ssl=1)

TestLodge is one of the best choices in terms of Test Management Tools. TestLodge supports you to create [test plans](https://www.softwaretestingmaterial.com/test-plan/), [test cases](https://www.softwaretestingmaterial.com/test-case-template-with-explanation/), test runs, [defects](https://www.softwaretestingmaterial.com/write-good-bug-report/), and [reporting](https://www.softwaretestingmaterial.com/generate-extent-reports/). Some of the TestLodge’s features that make it worth choosing this Test Management Tool are as follows

**Features:**

* Creating Test Plan
* Creating Test Suites
* Creating Test Cases
* Tracking Bugs
* Graphical Reporting System
* Integration with popular bug tracking tools

<https://www.softwaretestingmaterial.com/test-management-tools/>

## Decision Coverage

Decision coverage reports the true or false outcomes of each Boolean expression. In this coverage, expressions can sometimes get complicated. Therefore, it is very hard to achieve 100% coverage.

That's why there are many different methods of reporting this metric. All these methods focus on covering the most important combinations. It is very much similar to decision coverage, but it offers better sensitivity to control flow.

[https://www.guru99.com/images/1/102518_1122_CodeCoverag12.jpg](https://www.guru99.com/images/1/102518_1122_CodeCoverag12.jpg)

### Example of decision coverage

Consider the following code-

Demo(int a) {

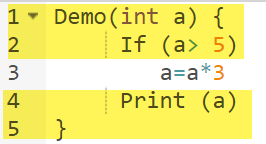
If (a> 5)

a=a\*3

Print (a)

}

alue of a is 2

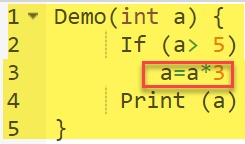
[](https://www.guru99.com/images/1/102518_1122_CodeCoverag8.png)

The code highlighted in yellow will be executed. Here the "No" outcome of the decision If (a>5) is checked.

Decision Coverage = 50%

**Scenario 2:**

Value of a is 6

[](https://www.guru99.com/images/1/102518_1122_CodeCoverag9.png)

The code highlighted in yellow will be executed. Here the "Yes" outcome of the decision If (a>5) is checked.

Decision Coverage = 50%

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Value of A | Output | Decision Coverage |
| 1 | 2 | 2 | 50% |
| 2 | 6 | 18 | 50% |

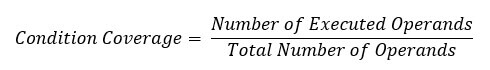
## Condition Coverage

Conditional coverage or expression coverage will reveal how the variables or subexpressions in the conditional statement are evaluated. In this coverage expressions with logical operands are only considered.

For example, if an expression has Boolean operations like AND, OR, XOR, which indicated total possibilities.

Conditional coverage offers better sensitivity to the control flow than decision coverage. Condition coverage does not give a guarantee about full decision coverage

The formula to calculate Condition Coverage:

[](https://www.guru99.com/images/1/102518_1122_CodeCoverag14.jpg)

Example:

[https://www.guru99.com/images/1/102518_1122_CodeCoverag11.png](https://www.guru99.com/images/1/102518_1122_CodeCoverag11.png)

For the above expression, we have 4 possible combinations

* TT
* FF
* TF
* FT

Consider the following input

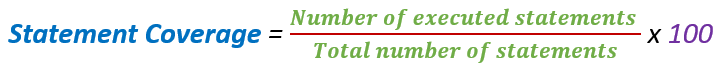
|  |  |  |  |
| --- | --- | --- | --- |
| X=3  Y=4 | (x<y) | TRUE | Condition Coverage is ¼ = 25% |
| A=3  B=4 | (a>b) | FALSE |

## Statement Coverage

What is Statement Coverage?

Statement coverage is a white box test design technique which involves execution of all the executable statements in the source code at least once. It is used to calculate and measure the number of statements in the source code which can be executed given the requirements.

Statement coverage is used to derive scenario based upon the structure of the code under test.

[](https://www.guru99.com/images/jsp/030116_0814_LearnStatem1.png)

In [White Box Testing](https://www.guru99.com/white-box-testing.html), the tester is concentrating on how the software works. In other words, the tester will be concentrating on the internal working of source code concerning control flow graphs or flow charts.

Generally in any software, if we look at the source code, there will be a wide variety of elements like operators, functions, looping, exceptional handlers, etc. Based on the input to the program, some of the code statements may not be executed. The goal of Statement coverage is to cover all the possible path's, line, and statement in the code.

Let's understand this with an example, how to calculate statement coverage.

Scenario to calculate Statement Coverage for given source code. Here we are taking two different scenarios to check the percentage of statement coverage for each scenario.

**Source Code:**

Prints (int a, int b) { ------------ Printsum is a function

int result = a+ b;

If (result> 0)

Print ("Positive", result)

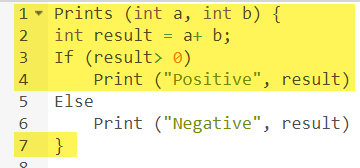
Else

Print ("Negative", result)

} ----------- End of the source code

**Scenario 1:**

If A = 3, B = 9

[](https://www.guru99.com/images/1/102518_1122_CodeCoverag2.png)

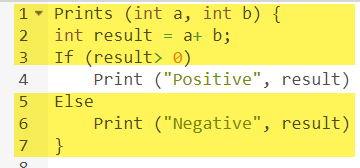
The statements marked in yellow color are those which are executed as per the scenario

Number of executed statements = 5, Total number of statements = 7

Statement Coverage: 5/7 = 71%

**Scenario 2:**

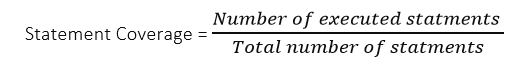
If A = -3, B = -9

[](https://www.guru99.com/images/1/102518_1122_CodeCoverag4.png)

The statements marked in yellow color are those which are executed as per the scenario.

Number of executed statements = 6

Total number of statements = 7

[](https://www.guru99.com/images/jsp/030116_0814_LearnStatem6.png)

Statement Coverage: 6/7 = 85%

[https://www.guru99.com/images/jsp/030116_0814_LearnStatem3.png](https://www.guru99.com/images/jsp/030116_0814_LearnStatem3.png)

But overall if you see, all the statements are being covered by 2nd scenario's considered. So we can conclude that overall statement coverage is 100%.

[https://www.guru99.com/images/jsp/030116_0814_LearnStatem4.png](https://www.guru99.com/images/jsp/030116_0814_LearnStatem4.png)

**What is covered by Statement Coverage?**

1. Unused Statements
2. Dead Code
3. Unused Branches