Manual Testing

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.

Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect. The developer fixed the defects and handed it to the tester for retesting.

Manual testing is mandatory for every newly developed software before automated testing. This testing requires great efforts and time, but it gives the surety of bug-free software. Manual Testing requires knowledge of manual testing techniques but not of any automated testing tool.

Manual testing is essential because one of the software testing fundamentals is "100% automation is not possible."

Why we need manual testing

Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it.

If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.

If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

Types of Manual Testing

There are various methods used for manual testing. Each technique is used according to its testing criteria. Types of manual testing are given below:

- White Box Testing
- Black Box Testing
- Gray Box Testing

White-box testing

The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing.

Black box testing

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

Gray Box testing

Gray box testing is a combination of white box and Black box testing. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing.

How to perform Manual Testing

- First, tester observes all documents related to software, to select testing areas.
- Tester analyses requirement documents to cover all requirements stated by the customer.
- Tester develops the test cases according to the requirement document.
- All test cases are executed manually by using Black box testing and white box testing.
- If bugs occurred then the testing team informs the development team.
- The Development team fixes bugs and handed software to the testing team for a retest.

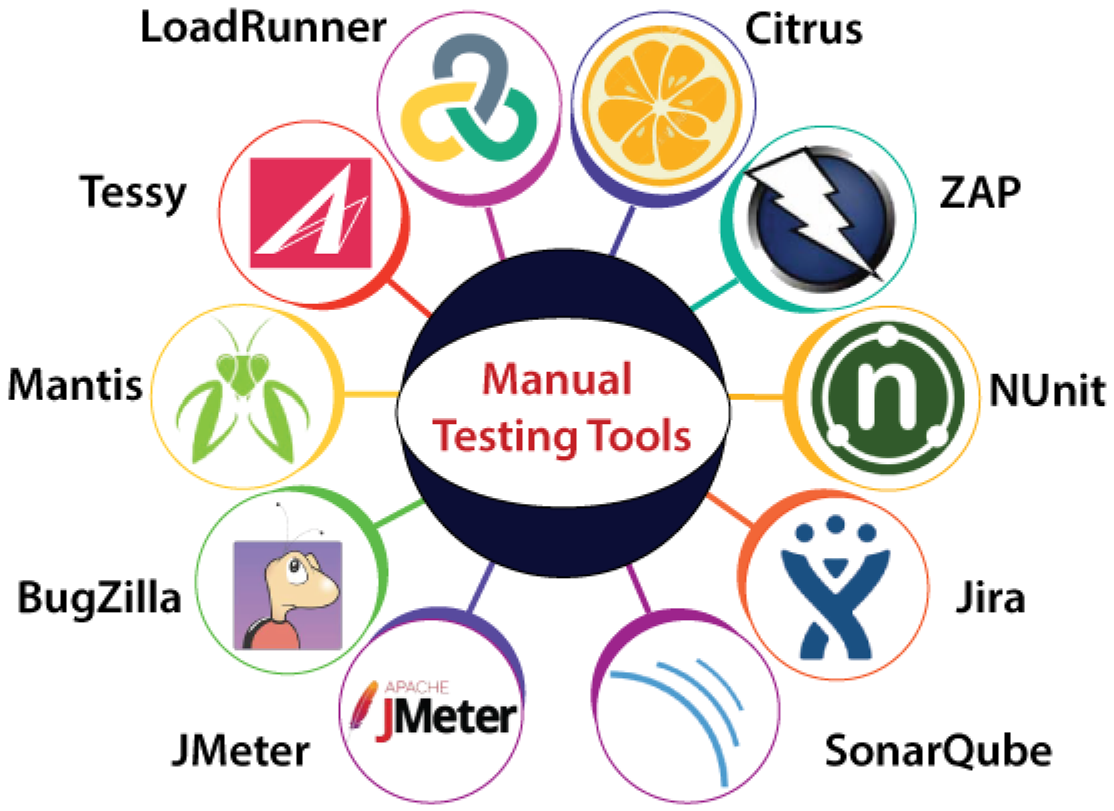
Advantages of Manual Testing

- It does not require programming knowledge while using the Black box method.
- It is used to test dynamically changing GUI designs.
- Tester interacts with software as a real user so that they are able to discover usability and user interface issues.

- It ensures that the software is a hundred percent bug-free.
- Easy to learn for new testers.

Manual testing tools

In manual testing, different types of testing like unit, integration, security, performance, and bug tracking, we have various tools such as Jira, Bugzilla, Mantis, Zap, NUnit, Tessy, LoadRunner, Citrus, SonarQube, etc. available in the market. Some of the tools are open-source, and some are commercial.



LoadRunner

It is most commonly used performance testing tools. LoadRunner is mainly used to support performance testing for the wide range of procedures, number of approaches, and application environments.

The main purpose of executing the LoadRunner tool is to classify the most common sources of performance issues quickly.

Citrus

Citrus is an integration testing tool, which is the most commonly used test framework. It is written in **Java programming** language. It is mostly used to request and respond to server-side and client-side and validate the XML JSON files.

ZAP

ZAP is an open-source web application security scanner. It is stands for **Zed Attack Proxy**. Just like some other tools, it is also written in the JAVA programming language. It is the most effective **Open Web Application Security Projects** [OWASP].

NUnit

NUnit is one of the most frequently used unit testing tools. It is an open-source tool and primarily derived from the **JUnit**.

It was completely written in the C# programming language and suitable for all .Net languages.

JIRA

The most regularly used bug tracking tool is **JIRA**, which is an open-source tool. It is used for bug tracking, project management, and issue tracking.

In this tool, we can easily track all kinds of bugs or defects related to the software and produced by the test engineers.

SonarQube

Another testing tool of manual testing is SonarQube, which improves our workflow with continuous code quality and code security. It is flexible with the use of plug-ins.

It is completely written in the JAVA programming language. It offers fully automated evaluation and integration with Ant, Maven, Gradle, MSBuild, and constant integration tools. SonarQube has the ability to record a metrics history and gives the evolution graph.

JMeter

JMeter is an open-source tool that is used to test the performance of both static and dynamic resources and dynamic web applications.

It is completely designed on the JAVA application to load the functional test behavior and measure the application's performance.

It facilitates users or developers to use the source code for the development of other applications.

Bugzilla

Another bug tracking tool used in manual testing is **Bugzilla**.

It is most widely used by many organizations to track the various bugs of the application.

Bugzilla is an open-source tool that helps the customer and the client to keep track of the defects. Bugzilla is also considered a test management tool because in this, we can easily link other test case management tools such as ALM, Quality Centre, etc.

Mantis

Mantis is a web-based bug tracking system. ManitsBT stands for **Mantis Bug Tracker**. It is used to follow the software defects and performed in the PHP programming language. It is also an open-source tool.

MANTIS

YOUR TAGLINE GOES HERE

Tessy

Another integration testing tool is **Tessy**, which is used to perform the integration and unit testing for the embedded software. It also helps us to discover the code coverage of the software or an application.

It can easily manage the entire test organization, including business needs, test management, coverage quantity, and traceability.

Automation Testing,

automation testing, which is used some specific tools to execute the test scripts without any human interference. It is the most acceptable way to enhance the efficiency, productivity, and test coverage of Software testing.

With the help of an automation testing tool, we can easily approach the test data, handle the test implementation, and compares the actual output against the expected outcome.

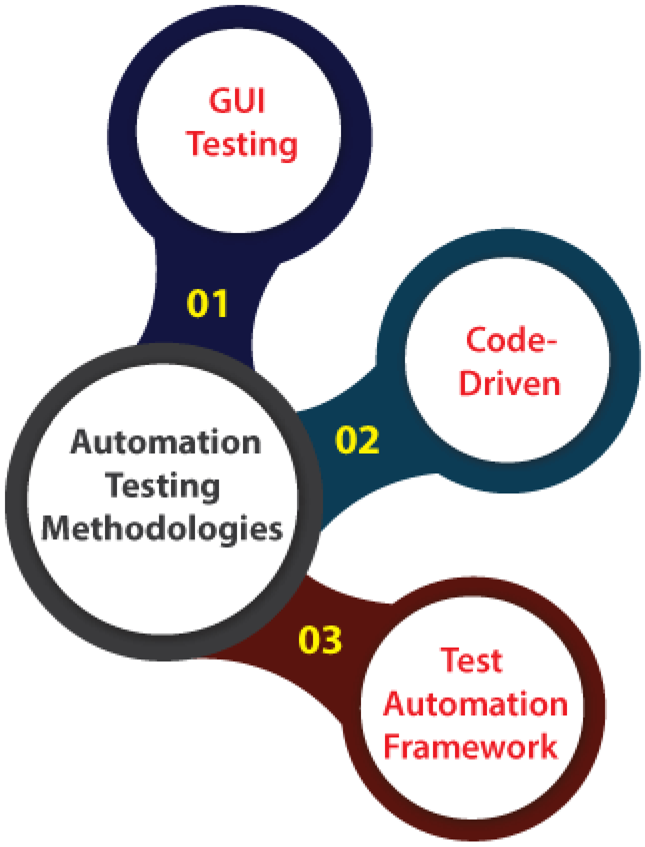
The execution of automation testing provides us various advantages, which are as discussed below:

- **Reusability**
- **Consistency**
- **Running tests anytime (24/7)**
- **Early Bug detection**
- **Less Human Resources**

Automation Testing Methodologies

Automation testing contains the following three different methodologies and approaches, which will help the test engineer to enhance the software product's quality.

- **GUI Testing**
- **Code-Driven**
- **Test Automation Framework**



Now, let's understand the different approaches of automation testing one by one:

1. GUI (Graphical user interface) Testing

In this approach, we can implement that software or an application, which contains GUIs. So, that the automation test engineers can record user actions and evaluate them many times.

We know that the **Test cases** can be written in several programming languages like JAVA, C#, Python, Perl, etc.

2. Code-Driven

The code-driven technique is the subsequent methodology used in automation testing. In this method, the test engineer will mainly concentrate on test case execution in order to identify whether the several parts of code are performing according to the given requirement or not.

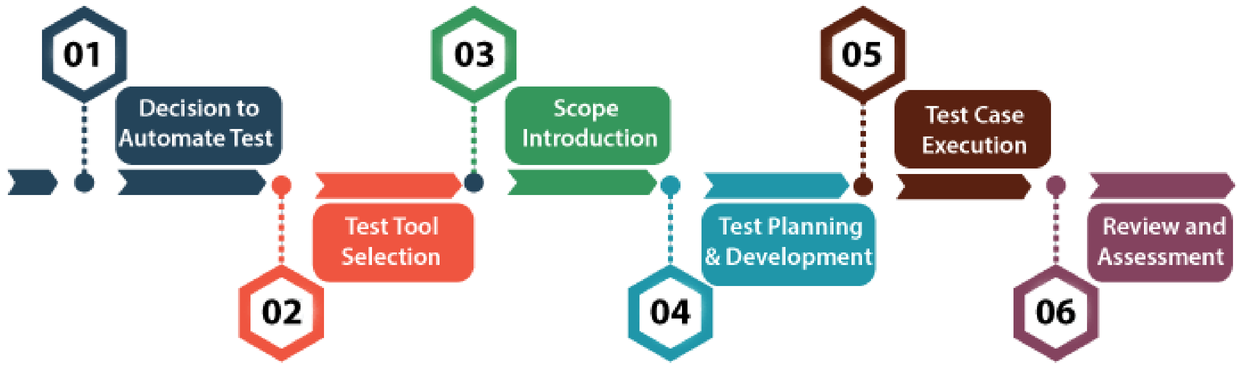
3. Test Automation Framework

Another approach in automation testing is **test automation framework**. The test automation framework is a set of rules used to generate valuable results of the automated testing activity.

Similarly, it brings together test data sources, function libraries, object details, and other reusable modules.

Automation Testing Process

The automation testing process is a systematic approach to organize and execute testing activities in a manner that provides maximum test coverage with limited resources. The structure of the test involves a multi-step process that supports the required, detailed and inter-related activities to perform the task.



Step1: Decision to Automation Testing

It is the first phase of **Automation Test Life-cycle Methodology (ATLM)**. At this phase, the main focus of the testing team is to manage expectations from the test and find out the potential benefits if applying the automated testing correctly.

Step2: Test Tool Selection

Test Tool Selection represents the second phase of the **Automation Test Life-cycle Methodology (ATLM)**. This phase guides the tester in the evaluation and selection of the testing tool.

Since the testing tool supports almost all testing requirements, the tester still needs to review the system engineering environment and other organizational needs and then make a list of evaluation parameters of the tools. Test engineers evaluate the equipment based on the provided sample criteria.

Step3: Scope Introduction

This phase represents the third phase of **Automation Test Life-cycle Methodology (ATLM)**. The scope of automation includes the testing area of the application.

Step4: Test Planning and Development

Test planning and development is the fourth and most important phase of Automation Test Life -cycle Methodology (ATLM) because all the testing strategies are defined here. Planning of long -lead test activities, the creation of standards and guidelines, an arrangement of the required combination of hardware, software and network to create a test environment, defect tracking procedure, guidelines to control test configuration and environment all are identified in this phase.

Step5: Test Case Execution

Test case Execution is the sixth phase of Automation Test Life -cycle Methodology (ATLM). It takes place after the successful completion of test planning. At this stage, the testing team defines test design and development. Now, test cases can be executed under product testing.

Step6: Review and Assessment

Review and assessment is the sixth and final stage of the automated testing life cycle but the activities of this phase are conducted throughout the whole life cycle to maintain continuous quality improvement. The improvement process is done via the evaluation of matrices, review and assessment of the activities.

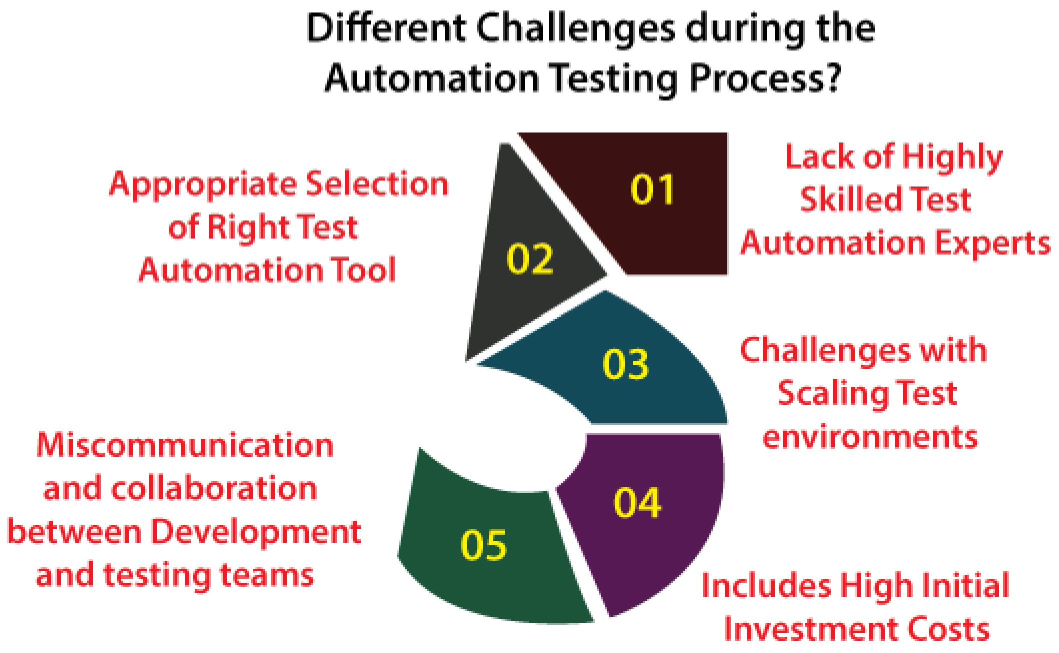
During the review, the examiner concentrates whether the particular metric satisfies the acceptance criteria or not, if yes, then it is ready to use in software production. It is comprehensive as test cases cover each feature of the application.

What are different challenges faced during the Automation testing process, and how to overcome them?

The **automation testing process** carries about large numbers of advantages for an organization. Usually, with automated testing, the testing team of the specified method of software validation can accomplish the amplified test coverage.

Though we may face various challenges during the automation testing process; therefore, we need a thorough follow-up process to reach successful test automation execution.

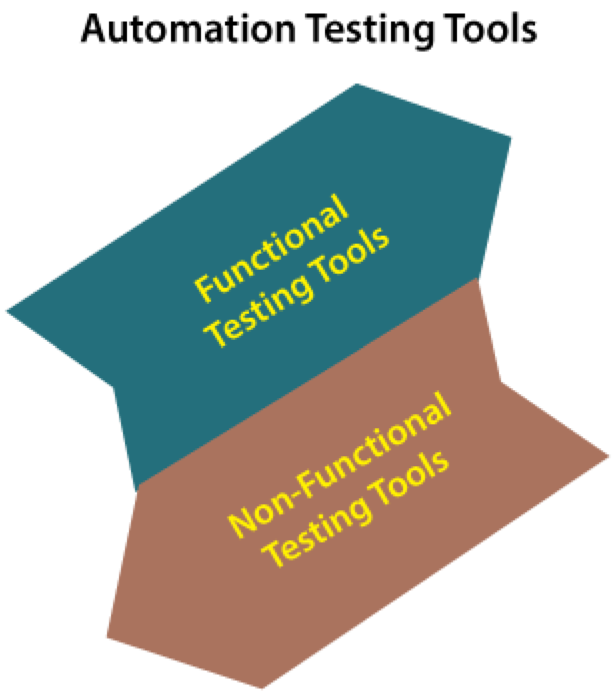
Some of the common challenges are as discuss below, which we might encounter throughout the test automation process of an application:


Different Challenges during the Automation Testing Process?
01
Lack of Highly Skilled Test Automation Experts
02
Appropriate Selection of Right Test Automation Tool
03
Challenges with Scaling Test environments
04
Includes High Initial Investment Costs
05
Miscommunication and collaboration between Development and testing teams

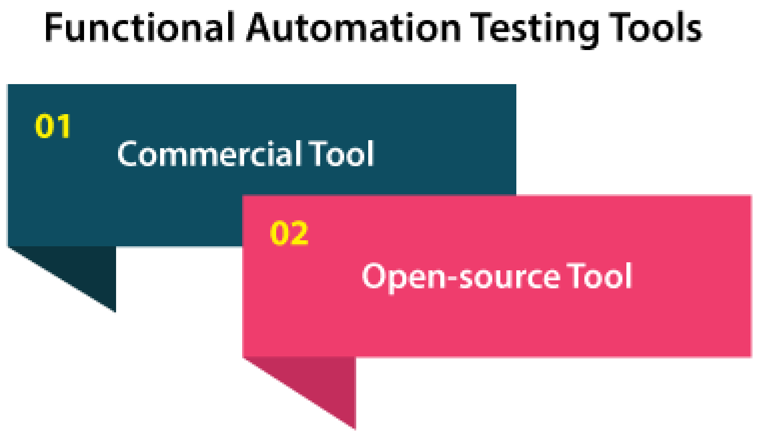
Automation Testing Tools

Automation testing tools can describe in two categories, which are as follows:

- Functional Testing Tools
- Non-Functional Testing Tools



- Commercial Tool
- Open-source Tool

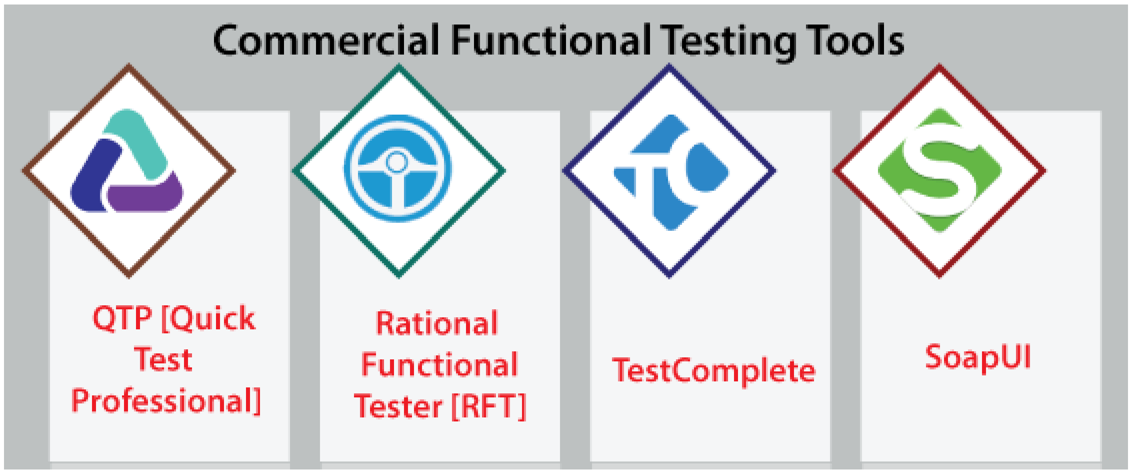


Commercial Functional Testing Tools

Commercial functional testing tools are those testing tools, which are not available freely in the market. These tools are also known as **licensed tools**. The licensed tools include various features and approaches as compared to the open-source tools.

Some of the most important commercial tools are as follows:

- **QTP[Quick Test Professional]**
- **Rational Functional Tester [RFT]**
- **TestComplete**
- **SoapUI**


Commercial Functional Testing Tools
QTP [Quick Test Professional]
Rational Functional Tester [RFT]
TestComplete
SoapUI

QTP

- QTP signify as **Quick Test Professional**, but now it is known as Micro Focus **UFT (Unified Functional Testing).**
- Mainly, it is used to automate the functional regression test cases of the web-based application or the software.
- In order to test the application, deploy the objects, and for analysis purposes, it is designed on the scripting languages such as **VBScript**.
- It follows the concept of **keyword-driven testing** for defining the test creation and maintenance.

- The new test engineers can easily use the QTP tool because it allows them to develop the test cases right from the application.

RFT

- RFT stands for **Rational Functional Tester,** which is used to execute the functional regression test cases.
- It follows the concept of Data-driven testing and GUI testing.
- It builds the automated functional tests by recording an end-user's activities on the system under test and repeating the actions on request to perform a test.

TestComplete

- Another functional automated testing tool is TestComplete, which acquires by **SmartBear Software**.
- TestComplete has some inbuild capability, which helps the test engineers to develop the automated tests for Microsoft Windows, Web, Android, and iOS applications.
- In this tool, the tests can be recorded, scripted, or manually created with keyword-driven processes and used for automatic playback and error logging.
- It includes the three different modules, **like Web, Desktop, and Mobile**, where all the modules involve the functionality to develop the automated tests on the particular platform.

SoapUI

- SoapUI is the most extensively used automation tool. It is mainly used to test the **web services**and **web APIs of SOAP** and **REST interfaces**.
- It allows the test engineers to test functional, regression testing, and other testing types on various Web services and APIs.
- SoapUI tool develops the mocks where test engineers can test real applications.
- It is entirely written on **JAVA** and Groovy programming languages.

Open-source Functional Testing Tools

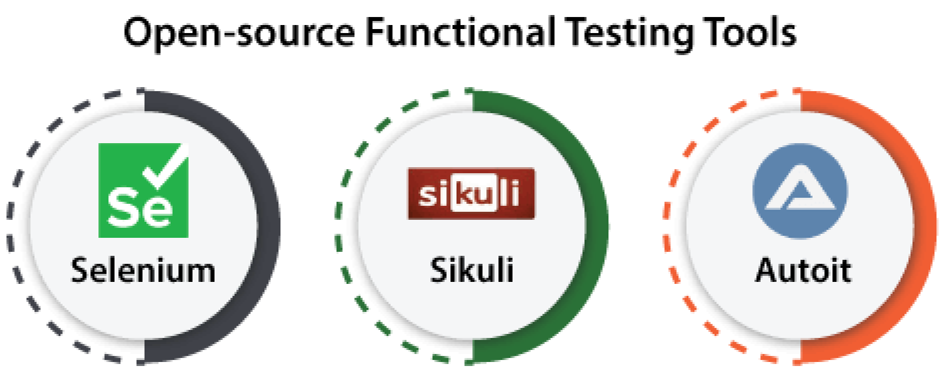
Open-source functional testing tools are those tools, which are available freely in the market. These tools have less functionality and features than the **commercial/licensed tools**, but sometimes working on the commercial tool became an expensive process.

That's why some well-known organizations preferred to use open-source tools.

Following are the most commonly used open-source functional automation testing tools:

- **Selenium**

- **Sikuli**
- **Autoit**



Selenium

Whenever we are talking about the open-source tools of automation testing, one name came into every automation test engineer's mind: Selenium.

- Selenium is the **highly recommended** and widely used functional testing tool, which is well-suited for non-functional testing tools.
- Selenium is an open-source tool which means, it does not require any licensing.
- We can only test the web application using the selenium tool, and the Stand-alone application cannot automate in Selenium.
- It is most commonly used to implement Functional test scripts.
- It can be combined with several automation testing tools to achieve continuous testing, for example, **Maven and Jenkins**.
- It can be associated with many devices such as **TestNG and JUnit** to manage the test cases and generate the test reports.

Sikuli

- Another open-source functional automation testing tool is **Sikuli**.
- It is a GUI based test automation tool, which can easily automate the Flash objects as most of the automation testing tool like selenium does not support the flash-object automation.
- Most commonly Sikuli is used to interact with fundamentals of web pages and control the windows-based popups.
- With the help of Sikuli tool, we can easily test the windows application.

AutoIt

- **AutoIt**is another open-source tool used for functional automation testing.
- It is a freeware scripting language designed to test the Windows GUI and general scripting.
- The AutoIt script is written in a primary language.
- It can replicate any grouping of keystrokes, mouse movement, and window or control operation.

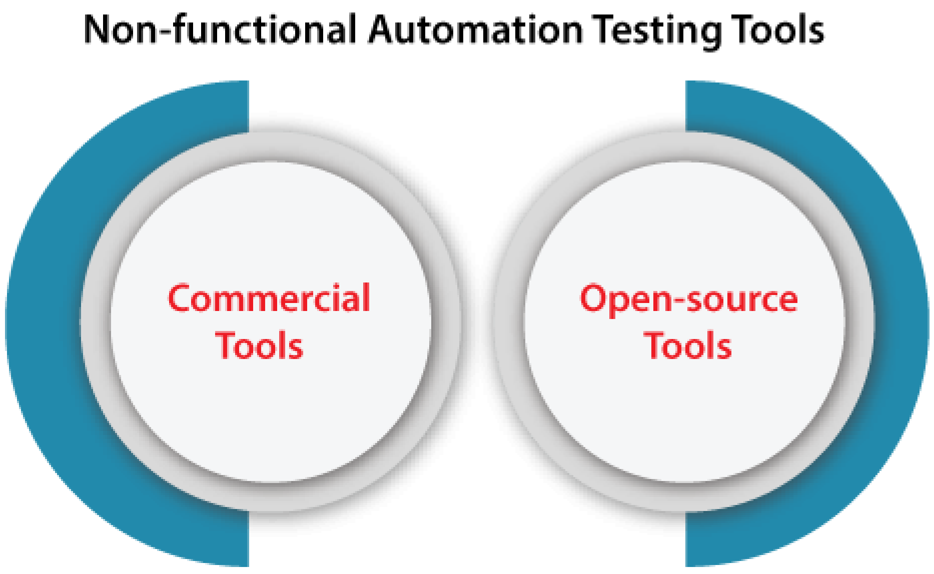
Non-functional Automation Testing Tools

The automation test engineer uses the non-functional automation testing tool to execute the non-functional performance **test cases.**

For example, testing the application's response time under significant load, let say, 100 users.

Just like functional automation testing tools, the non-functional automation testing tools divided into two different categories, which are as below:

1. **Commercial Tools**
2. **Open-source Tools**



Commercial Non-Functional Automation Testing Tools

These are those tools, which cannot be used freely, as they required a proper license. The commercial tools have additional features and functionality as compared to the other open-source testing tools.

The usage of these types of tools helps us to enhance the efficiency of the software product.

Let's see some of the most commonly used commercial non-functional automation testing tools.

- **LoadRunner**
- **Silk Performer**

LoadRunner [HP Performance Tester]

LoadRunner is one of the most popular non-functional tools. It is mainly used to support performance testing for a wide range of protocols, several technologies, and application environments. LoadRunner is the licensed tool.

Silk Performer

Another non-functional automation testing tool is Silk Performer. It can test the various application environments with thousands of simultaneous users.

It makes sure that applications and server up times are sustained when faced with the highest customer usage.

It is one of the most commonly used enterprise-class load and stress testing tools, supporting an extensive range of protocols.

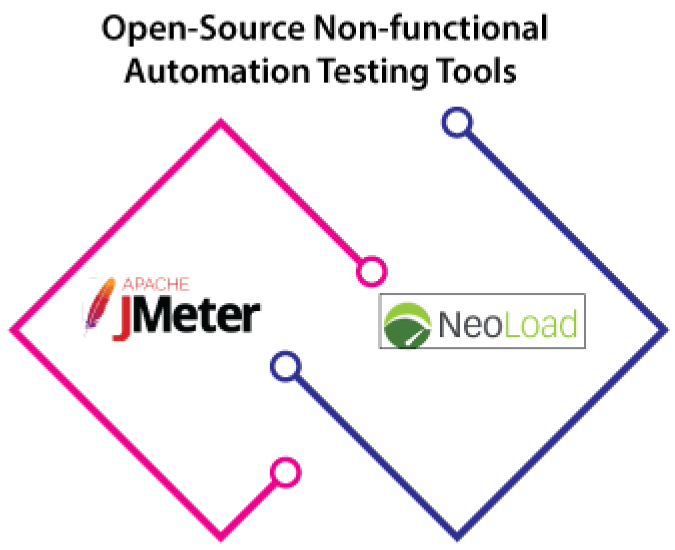
Open-Source Non-functional Automation Testing Tools

The open-source non-functional automation testing tools can be used easily as they are freely available in the market. These tools have less functionality as compared to the commercial testing tools.

But because they are open-source and used quickly, some organizations prefer to use these tools as they do not require any cost.

Some of the most commonly used open-source non-functional automation testing tools are as follows:

- **JMeter**
- **NeoLoad**



JMeter

JMeter is one of the most significantly used open-source non-functional automation testing tools. JMeter is entirely designed on the JAVA application to load the efficient test performance and measure an application's productivity.

It is mainly simplifying users or developers to use the source code for the enhancement of other applications. It is used to test the implementation of both static and dynamic resources and dynamic web applications.

NeoLoad

Another most commonly used open-source tool in automation testing is NeoLoad and **Neotys** develop it.

It is used to test the performance test scenarios and also helps us to identify the bottleneck areas in the web and the mobile application development process.

It is faster as compared to other traditional tools. NeoLoad will support a wide range of web, mobile, and packaged applications, such as **SAP, Oracle, Salesforce**, etc., that cover all our testing needs.

Advantages of Automation Testing

- Automation testing takes less time than manual testing.
- A tester can test the response of the software if the execution of the same operation is repeated several times.
- Automation Testing provides re-usability of test cases on testing of different versions of the same software.
- Automation testing is reliable as it eliminates hidden errors by executing test cases again in the same way.
- Automation Testing is comprehensive as test cases cover each and every feature of the application.
- It does not require many human resources, instead of writing test cases and testing them manually, they need an automation testing engineer to run them.
- The cost of automation testing is less than manual testing because it requires a few human resources.

Manual Testing vs. Automated Testing

Aspect of Testing	**Manual**	**Automation**
Test Execution	Done manually by QA testers	Done automatically using automation tools and scripts
Test Efficiency	Time-consuming and less efficient	More testing in less time and greater efficiency
Types of Tasks	Entirely manual tasks	Most tasks can be automated, including real user simulations
Test Coverage	Difficult to ensure sufficient test coverage	Easy to ensure greater test coverage