Preface

Repeatable test cases are executed again and again in the test life cycle. E.g. login test is manually repeated again and again with different inputs. E.g. login using emails is tested with different set of inputs because email validation requires a lot of constraints so these types of test need a lot of efforts if test manually …furthermore regression tests requires a lot of repetition i.e. test after tests… similarly combinatorial testing requires a lot of manual efforts i.e a lot of test cases to validate the component. Similarly coverage testing in white box needs a lot of efforts if erform manually .. Hence theses types of test should perform with testing tool to reduce avoid manual efforts. But every test should not be done automatically.. e.g. loading a pdf file. This type of test should be performed once to test whether the uploading component is working properly. So these test are not repeated so do not erform automatically because automatic testing requires resources . i.e it need test planning , test tool requirements etc .

**Manual Vs Automated Testing**

If you execute the test cases manually without any tool support, it is known as manual testing. It is time consuming and less reliable. E.g. you are testing login by manually inputting the login id n password again and again.

If you execute the test cases by tool support, it is known as automated testing. It is fast and more reliable. E.g. if you set many types of input in the tool and tool is running the login component and giving it the inputs iteratively with out human effort.

In manual testing, a human performs the tests step by step, without test scripts. In automated testing, tests are executed automatically via test automation frameworks. That means automated testing is faster than manual testing.

The biggest **difference between manual and automation testing** is who executes the test case. In manual testing, the human tester does it. In automation testing, the tool does it but with the help of a tester.

**What Is Automated Testing?**

Automated testing is the process in which testers utilize tools and scripts to automate testing efforts.

Automated testing helps testers execute more test cases and improve test coverage. When comparing manual vs. automation testing, manual takes longer.

Automated testing is a software testing technique that automates the process of validating the functionality of software and ensures it meets requirements before being released into production. With automated testing, an organization can run specific software tests at a faster pace without human testers. Automated testing is best suited for large number or repetitive test cases.

Automated software testing uses scripted sequences executed by testing tools. Automated testing tools examine the software, report outcomes and compare results with earlier test runs. An automated test [script](https://www.techtarget.com/whatis/definition/script) can be created once and then used repeatedly.

An [organization can apply automated tests](https://www.techtarget.com/searchsoftwarequality/A-comprehensive-test-automation-guide-for-IT-teams) to a broad range of cases, such as unit, application programming interface ([API](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API)) and regression testing. The main benefit of automated software testing is that it simplifies much of the manual effort into a set of scripts. For example, if unit testing consumes a large percentage of a quality assurance ([QA](https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance)) team's resources, then this process should be evaluated as a candidate for automation.

Automated tests can run repeatedly at any time of day and are an extremely important part of continuous testing, continuous integration ([CI](https://www.techtarget.com/searchsoftwarequality/definition/continuous-integration)) and continuous delivery ([CD](https://www.techtarget.com/searchitoperations/definition/continuous-delivery-CD)) software.

Automated testing is more efficient.

|  |  |  |
| --- | --- | --- |
| **Manual Testing vs. Automation 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 |

Tests that are normally automated include the following:

* Acceptance tests.
* API tests
* Integration tests.
* Regression tests.
* System tests.
* Unit tests.

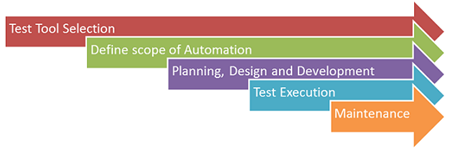
## Benefits of automated testing

Automated testing can boost a QA team's efficiency. Benefits of automating the testing process include the following:

* Better reporting capabilities.
* More frequent tests.
* Enhanced resource efficiency.
* Faster than manual testing and has a faster feedback cycle
* Higher accuracy.
* Improved bug detection.
* Improved [return on investment](https://www.techtarget.com/searchcio/definition/ROI) over manual testing.
* Increased coverage.
* Reusable test scripts.
* Scalability.

## **Automated Testing Process:**

Following steps are followed in an Automation Process



Test Automation Process

**Step 1)** Test Tool Selection

**Step 2)** Define scope of Automation

**Step 3)** Planning, Design and Development

**Step 4)** Test Execution

**Step 5)** Maintenance

### Step 1) Test tool selection

Test Tool selection largely depends on the technology the Application Under Test is built on. For instance, QTP does not support Informatica. So QTP cannot be used for testing Informatica applications.

### Step 2) Define the scope of Automation

The scope of automation is the area of your Application Under Test which will be automated. Following points help determine scope:

* The features that are important for the business
* Scenarios which have **a large amount of data**
* **Common functionalities** across applications
* **The complexity** of test cases
* Ability to use the same test cases for cross-browser testing

### Step 3) Planning, Design, and Development

During this phase, you create an Automation strategy & plan, which contains the following details-

* Automation tools selected
* Framework design and its features
* In-Scope and Out-of-scope items of automation
* Automation testbed preparation
* Schedule and Timeline of scripting and execution
* Deliverables of Automation Testing

### Step 4) Test Execution

Automation Scripts are executed during this phase. The scripts need input test data before there are set to run. Once executed they provide detailed test reports.

Execution can be performed using the automation tool directly or through the Test Management tool which will invoke the automation tool.

### Step 5) Test Automation Maintenance Approach

**Test Automation Maintenance Approach** is an automation testing phase carried out to test whether the new functionalities added to the software are working fine or not. Maintenance in automation testing is executed when new automation scripts are added and need to be reviewed and maintained in order to improve the effectiveness of automation scripts with each successive release cycle.

**How to Choose Between Manual vs. Automation Testing**

The key characteristics of manual testing include:

1. **Flexible:** Manual testers can dynamically change their approach as a software product changes. They can perform random testing and seek out bugs that are not already outlined in test cases.
2. **Responsive:** Exploratory manual testing can provide critical feedback in the first stages of the software development life cycle.

**1) When flexibility is needed**

Although [manual testing](https://smartbear.com/blog/why-manual-testing-is-still-important/) is not always as accurate as automation, manual processes allow testers to have more flexibility in their operations. Because automated testing is inherently useful for repeated cases, this often means that the same code and scripts will be leveraged every time. If testers have an idea they want to work on right away, however, doing it with automation requires establishing test cases, programming them with the automation tool and then running the test, which takes a lot of time, according to Base36. In this case, it would be easier - and faster - to execute through manual operations.

"With manual testing, you can just quickly test and see the results," Base36 stated. "Automatic tests take more time to set up, which doesn't allow you to test ideas quickly and easily."

**3) When short-term projects are active**

Automation requires a lot of investment and planning, both of which can be extraneous in short-term projects. In an interview with TechTarget, John Overbaugh, senior SDET lead at Microsoft, stated that [automated testing](http://searchsoftwarequality.techtarget.com/feature/When-to-use-manual-vs-automated-software-testing-tools) doesn't make sense in this case because of the costs. The upfront price of automating a project like this could be too high for the return on investment and value offered by the finished product. For this instance, manual testing would be cheaper and more beneficial overall to the bottom line.

"Automated testing doesn't make sense on short-term projects where the expense of setting up automation exceeds the value," Overbaugh said. "I have been involved in small projects that added a minor feature set with little or no code shared with other portions of the project and which would, in all likelihood, never be iterated on."

**4) When usability is being tested**

The success or failure of an application is largely dependent on its usability. However, this extends far beyond just running automated tests to ensure that no defects exist. Computers do not have original thought - they run actions that are programmed and cannot provide the type of feedback that a human user would give when using an app. For example, automated tests can easily find errors in code, but it could not tell how users would interact with a certain feature or how well people will be able to navigate the program.

If users find that a tool in the software does not work the way they need it to, this could be a critical factor in determining whether others adopt the app. This is the type of information that manual testing would generate. Software development consultant Kevlin Henney noted that exploratory and usability testing will both need human perceptions as machines "are great at faultless repetition, but less effective at exploring and following hunches." This could help quickly catch and mitigate defects that emerge when users engage with the app in certain ways.

**Automation Myths**

#### **Myth #1: Test automation is expensive and requires a lot of resources.**

**Reality:**While it’s  true that there’s an upfront investment required for test automation tools and resources, the benefits of automation far outweigh the costs. Automated testing saves major time, which means it reduces labor costs. It also ensures you don’t have an expensive disruption in service by improving the overall quality of software. Gartner reports that system downtime costs an average of $300,000 per hour. That means just one instance of downtime due to poor testing could put you in a financial [hole.](https://www.opkey.com/ebook/how-to-save-money-with-test-automation)

#### **Myth #2: Test automation totally eliminates the need for manual testing and replaces jobs.**

**Reality:** Test automation is not a replacement for manual testing. Automated testing is best used to complement manual testing, not replace it entirely. While automated testing can help speed up the testing process and catch certain types of defects, it cannot replace the human judgment that is needed for certain types of testing (such as exploratory testing).

## **Automation will replace testers??**

## The idea that you can take a tester with critical thinking skills, classically trained testing ability, and investigatory powers and get rid of them by putting code in their place is a fallacy. Scripts are not going to think critically or perform deep-dive investigations into your software. The scripts you employ are only as good as the tester who writes them.

## We no longer see the industry as made up of manual and automated testers.  We have testers with coding skills and those without – all contribute in different ways. Trained testers will always have a place because of their ability to think critically, beyond a script.

## Until we have powerful and affordable artificial intelligence, where the scripts are writing themselves, automated testing is just another tool in the box. It’s also worth noting that we fundamentally write software for humans to use, so a tester’s perspective will always be valuable.

#### **Myth #3: Test automation is only suitable for large projects with ample resources.**

**Reality:**Test automation works for any software or digital ecosystem. Automation can be used on projects of any size, and it can be especially useful for smaller projects that have limited resources and tight deadlines. Opkey provides testing for all sorts of custom applications, mobile interfaces, and websites. (In addition to our expertise 15+ packaged apps, including [Oracle Cloud](https://www.opkey.com/oracle), [Oracle EBS](https://www.opkey.com/oracle-ebs-automation), [Salesforce](https://www.opkey.com/salesforce), [SAP](https://www.opkey.com/sap-test-automation), and [Workday](https://www.opkey.com/workday-automation).)

#### **Myth #6: Test automation can be done by absolutely anyone.**

**Reality:** Let’s not swing too far to the other end of the pendulum. Test automation does require an understanding of the layout, metrics, and functions of your ERP ecosystem. If a user doesn’t have this basic knowledge, they may not be able to correctly interpret the results. They don’t need to know how to code, but they do need to be familiar with your packaged app.

#### **Myth #8: Test automation is a one-time activity.**

**Reality:**Test automation is a routine need for your system, like upkeeping a car. Changes, updates, and the natural growth of a business require systems to expand and be tested in new ways. Automated tests must be reviewed regularly to ensure they are still relevant and effective.

#### **Myth #10: Test automation is inflexible and cannot adapt to changes in software requirements. It is hard to maintain.**

**Reality:** Test automation can change with your system, faster than manual methods can. Automated tests are designed to be flexible and adaptable to changes in software and business requirements. With the right tools and processes in place, automated tests can be updated quickly and easily, often at the single click of a button.

## **Manual Testing vs Automated Testing – Key Differences**

**The difference between manual testing and automated testing is that in manual testing, you perform the tests step by step without the help of tools, whereas in automated testing, tests are executed automatically using automation tools & frameworks.**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Automation Testing** | **Manual Testing** |
| Speed | Automation Testing is faster than a manual testing. | Manual testing requires human resources and is time consuming as well. |
| Exploratory Testing | Automation will not allow any random testing. | It is possible in case of Manual Testing |
| Investment | The initial investment for automation testing is higher. | The initial investment for manual testing is lower. |
| Reliability | Automation Testing is more reliable, as it is performed by tools and scripts. | Manual testing won’t be as accurate as there is a possibility of the human error. |
| Investment Type | Investment is needed for testing tools and automation engineers. | Investment is needed for human resources. |
| Cost-effective | Automation Testing is cost effective. | Manual Testing is not cost effective as ROI is lower. |
| Test Reports | In automation testing, all stakeholders check test execution results by logging into automation system. | Manual Tests are usually updated in an Excel/Word, and test results are not readily available. |
| Performance Testing | Performance Tests like Stress Testing, Load Testing, Spike Testing, etc. have to be done by an automation tool only. | Performance Testing is difficult to do manually. |
| Set up | It needs less complex test execution set up. | Manual testing needs have a straightforward test execution setup. |
| Deadlines | Automated Tests will not have risks of missing out any test deadlines. | Manual Testing has a higher chances of missing out the test deadlines. |
| Framework | Automation testing employs frameworks like Data Driven, Keyword, and Hybrid to speed up the automation process. | Manual Testing won’t have frameworks but may use guidelines, checklists, and strict processes. |
| When to Use? | It is well suited for Regression Testing, Load Testing, Performance Testing, or repeatable functional test cases. | It will be suitable for Usability, Exploratory, and Adhoc Testing. |

|  |  |  |
| --- | --- | --- |
| **Testing Aspect** | **Manual Testing** | **Automated Testing** |
| **Accuracy** | Less accurate due to the higher possibility of human errors. However, it handles complex tests that require human thinking and judgment better. | Very accurate for repeatable, non-flaky test scenarios. However, it can be less accurate for tests that need human thinking and judgment or interactions that rely on integrated modules or systems. Poor design of test cases and errors in the test scripts might also lead to lower accuracy. |
| **Cost Efficiency** | Cost-effective for more complex tests, tests that involve some kind of investigation or judgment or usability, or tests that you will not run very often. | Cost-effective for predictable tests that you will repeat many times across multiple test cycles, especially regression tests. |
| **Reliability** | More reliable for performing exploratory testing and for identifying subtle issues or inconsistencies. | More reliable for repetitive tests. |
| **Test Coverage** | Covers a wide range of test scenarios, including tests that need human intervention. However, manual testing is not efficient when testing large and complex scenarios. | Covers a wider range of scenarios, such as large, complex, and repetitive ones. Yet, it is not suited for instances that need human intuition and judgment. |
| **Scalability** | Less efficient and more time-consuming. But more effective for UI-related and human instinct-needed tests | More efficient and effective for testing large, routine, and repetitive tasks at scale. |
| **Test Cycle Time** | Shorter turnaround time due to automatically performing tests and generating reports. However, setting up the environment and writing test scripts may take more time at the initial stage. | Effective in testing the user experience because UX typically involves judging the feeling or perception a user might have about the overall user-friendliness using multiple. |
| **User Experience** | Effective in testing the user experience because UX typically involves judging the feeling or perception a user might have about the overall user-friendliness using multiple . | Ineffective in testing the user experience of an application. |
| **Human Resources / Skills** | Does not require programming skills as it does not involve test scripts, but requires hands-on testing experience. | Typically requires programming skills. A strong understanding of at least one of the most common programming languages, like Python, Java, JavaScript, or C#, is recommended. Some modern test automation tools do not require as much scripting of code, although you may need to make sure those tools can reliably handle the tests you need to run based on the type of application you are testing. |

## **When To Use Manual Testing?**

Manual testing is only ideal when the scenario is complex and non-repetitive, making investing in automation for them time-consuming. Another scenario is when testers want to evaluate the application from the end user’s perspective, providing a more genuine and human feedback to the development team.

***In fact, we should not consider the two testing approaches as alternatives. It is significant to apply automation and manual testing correctly in your test plan.***

***Testing vs Debugging***

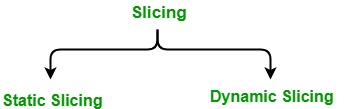
Testing is finding defects while debugging is locating their cause in the code and fixing them.

**What Testing activities can be automated?**

* Test Planning? Done manually.
* Test Case Preparation? Only through Model based testing and through A.I & M.L
* Test Case execution? This is the main candidate of automation.
* Analysis & Report? Reports are generated but analysis is typically limited and depends upon the expertise of Tester.

**Slicing**

**Slicing** or program slicing is a technique used in software testing which takes a slice or a group of program statements in the program for testing particular test conditions or cases that may affect a value at a particular point of interest. It can also be used for the purpose of debugging in order to find the bugs more easily and quickly.



**1. Static slicing:** 

* A static slice of a program contains all statements that may affect the value of a variable at any point for any arbitrary execution of the program.
* Static slices are generally larger.
* It considers every possible execution of the program.

**2. Dynamic slicing:** 

* A dynamic slice of a program contains all the statements that actually affect the value of a variable at any point for a particular execution of the program.
* Dynamic slices are generally smaller.
* Considers only a particular execution of the program.

Let us see an example by the given code snippet:

**int** z = 10;

**int** n;

cin >> n;

**int** sum = 0;

**if** (n > 10)

    sum = sum + n;

**else**

    sum = sum - n;

cout<<sum;

Static slice for the variable **sum**:

**int** n;

cin >> n;

**int** sum = 0;

**if** (n > 10)

    sum = sum + n;

**else**

    sum = sum - n;

Dynamic slice for the variable **sum** when n = 22;

**int** n;

cin >> n;

**int** sum = 0;

**if** (n > 10)

    sum = sum + n;

As it can be observed in the above example the static slice takes all the possible execution (in this case it is 2) of the program which may affect the value of the variable **sum**. Whereas in the case of dynamic slicing, it considers only a particular execution (when n = 22) of the program which actually affects the value of the variable **sum**.   
Hence, the dynamic slice is always smaller than a static slice.

Since, at the statement cout<<sum, the value of sum is not dependent on the statement itself hence it is not included in the slice.