# CHƯƠNG 6:

# AUTOMATION TESTING

Bộ môn Công Nghệ Phần Mềm
Khoa CNTT – Trường ĐH Ngoại Ngữ Tin Học TP.HCM

# OBJECTIVES

HELP STUDENTS:

UNDERSTANDING:

- What/why/when/who/how to do automation testing
- Requirements for automated software testing
- Automated testing process
- Test tool selection
- Automation testing frameworks

# WHAT?

- **Automation testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, manual testing is performed by a human sitting in front of a computer carefully executing the test steps.
- The automation testing software can also enter test data into the system under test, compare expected and actual results and generate detailed test reports. Software test automation demands considerable investments of money and resources.

# Why Test Automation?

- **Test automation** is the best way to increase the effectiveness, test coverage, and execution speed in software testing. Automated software testing is important due to the following reasons:
- Manual testing of all workflows, all fields, all negative scenarios is time and money consuming
- It is difficult to test for multilingual sites manually
- Test automation in software testing does not require human intervention. You can run automated test unattended (overnight)
- Test automation increases the speed of test execution
- Automation helps increase test coverage
- Manual testing can become boring and hence error-prone.

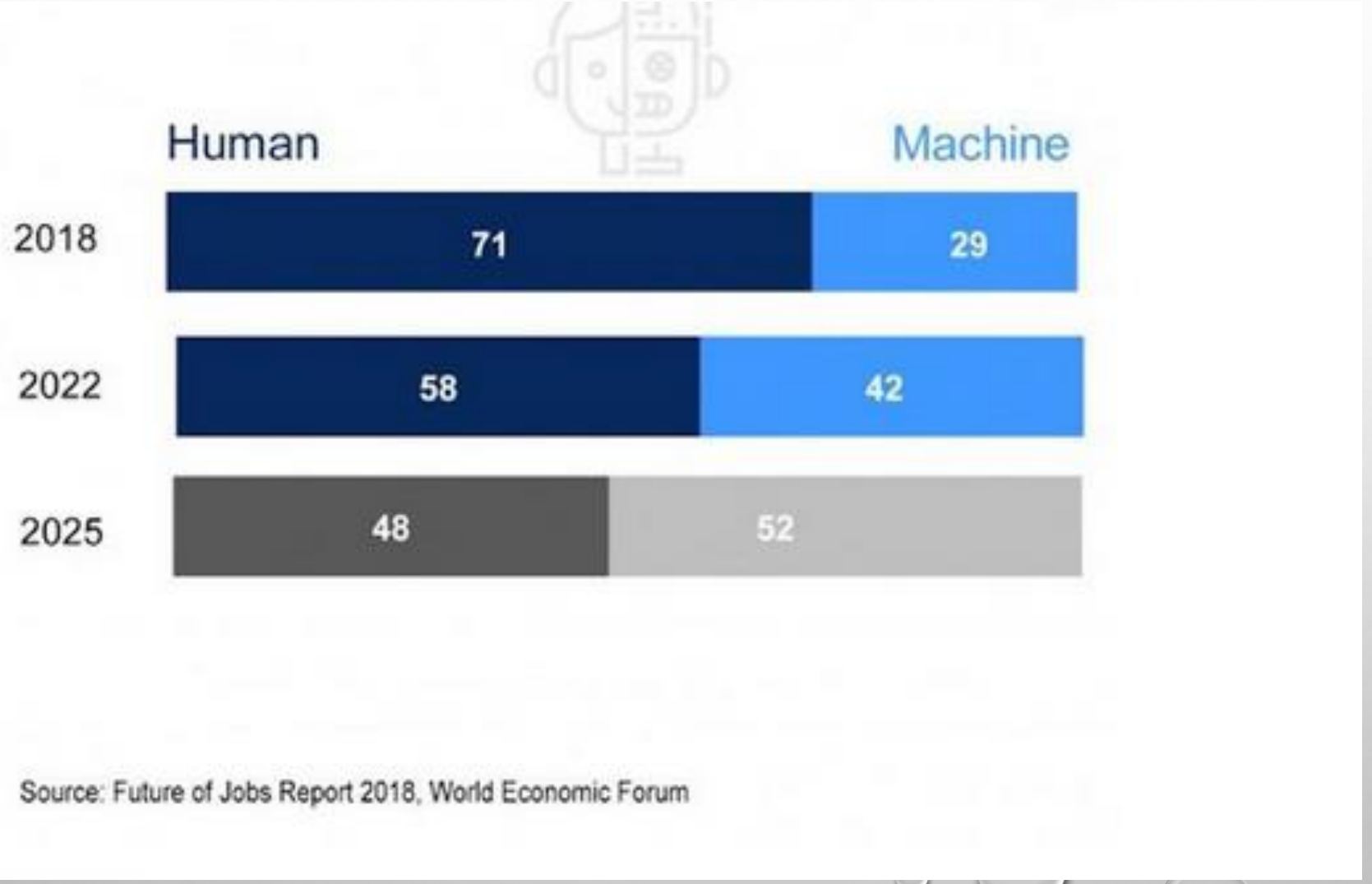
# What are the benefits of automation testing?

- Software testing has many benefits, which is why saas businesses across the globe are utilizing automation technology. Here are some of the biggest benefits of using automation testing for software development:
- **Detailed reporting capabilities** - automation testing uses well-crafted test cases for various scenarios. These scripted sequences can be incredibly in-depth, and provide detailed reports that simply wouldn't be possible when done by a human. Not to mention providing them in a shorter amount of time.
- **Improved bug detection** - one of the main reasons to test a product is to detect bugs and other defects. Automation testing makes this process an easier one. It's also able to analyze a wider test coverage than humans may be able to.

# What are the benefits of automation testing?

- **Simplifies testing** - testing is a routine part of the operations of most saas and tech companies. Making it as simple as possible is key. Using automation is extremely beneficial. When automating test tools, the test scripts can be reused. Manual testing, meanwhile, calls for a single code line to be written for the same test case, each time it needs to be run
- **Speeds up the testing process** - machines and automated technology work faster than humans. Along with improved accuracy, this is why we use them. In turn, this shortens your software development cycles.
- **Reduces human intervention** - tests can be run at any time of day, even overnight, without needing humans to oversee it. Plus, when it's conducted automatically, this can also reduce the risk of human error.
- **Saves time and money** - testing can be time-consuming. Though automation may require an initial investment, it can save money in the long run to become more cost-effective. Team members use their time in other areas and are no longer required to carry out manual testing in many situations. This improves their workflow.

# Rate of automation test



# When Should we Automate Your Software Testing?

- **WHEN THE COST MAKES SENSE**

*“Automation is never cheap. In some cases, you will need to use paid tools, and in others, you will need to hire someone just to write automation scripts. Generally, testing automation doesn’t make sense for simple projects so it’s better to use manual testing.*

*An overly-generalized rule of thumb for software automation is whether you value short-term costs or long-term roi. If there is currently no room in your budget, skip on testing automation, especially if it’s a smaller-scale project. If it’s a large, long-term project, automation can save time, frustration, and even money!”*

**- Michal kowalkowski, ceo and founder of nospoilers.Ai**

# When Should we Automate Your Software Testing?

- **WHEN YOU ARE USING REPETITIVE TESTS**

*"If you have a large amount or multiple repetitive software tests to do, automation is often the best option to save on time and get more in-depth reliable results."*

- **James boatwright, ceo of code galaxy**

*"The benefit of automated testing is linked to how many times a given test can be repeated, making it ideal for a system that is continuously running, with the opportunity to spot any issues as they arise, rather than reacting to ones that are already receiving complaints and losing you users."*

- **- Adam korbl, ceo & founder at ifax**

# When Should we Automate Your Software Testing?

## WHEN TIME WILL BE SAVED

*“Automated testing frees up time for your developers and QA team. This allows them to invest their time into other features which can help reduce costs. By freeing up your dev team’s time, they will also be motivated to contribute more and work on features that they’re really passionate about.*

*It also saves money as there is a less likely chance that regression bugs will be introduced into the system. Debugging is one of the most time-consuming tasks that a developer is responsible for. Automated testing helps to reduce cases where debugging is required.*

**- Husam machlovi, founder, managing partner and with pulp**

# When Should we Automate Your Software Testing?

## WHEN QUALITY IS SURE TO BE IMPROVED

Automation removes the possibility of human error. for that reason, in some scenarios quality can be dramatically improved by the use of automated testing. but you can also run hundreds of tests at once, meaning you will deliver a well-tested product that can be re-tested time and time again.

## WHEN TESTS ARE RUN FREQUENTLY

Having the capacity to run frequent tests at high volume just isn't a possibility in many engineering teams. Manual testing can only go so far, especially for smaller teams with no in-house testing team. If you would like to scale up your testing capacity, speak to one of our growth experts today to see what we can do for your team.

## WHEN YOU NEED TO RUN MULTIPLE TESTS AT ONCE

Running the same manual tests simultaneously is no easy feat. The likelihood of a team having the capacity to run 100 tests at exactly the same time is low. Automation makes this task super fast and enables teams to test rapidly without feeling the pressure.

# Automation testing requirements

1. Proficiency in programming languages. ...
2. **EXPERIENCE IN MANUAL TESTING**
3. Sufficient knowledge of automation tools. ...
4. Clearly understanding business requirements. ...
5. Well versed with test management tools. ...
6. Familarity with agile, devops methodologies & continuous delivery.
7. Experience with reporting, time management, analytical and communication capabilities

# Who ?

- **Test automation test engineer**

Test automation engineers design and develop programs that conduct automatic tests on new and existent software. They also use guidelines to develop programs and write automation scripts.

- **Responsibilities:**
  - Designing and developing test automation scripts.
  - Using test automation guidelines.
  - Researching issues in software through testing.
  - Collaborating with qa analysts and software developers to develop solutions.
  - Keeping updated with the latest industry developments.

# What are the types of automation testing?

- There are five key types of automation testing. Each can be used in different circumstances, depending on the application under test. You can analyze each one to see which would be best for you, or you can test them out with a trial. This is sometimes the best way to know which tool you should be using. Here are the main types of automation testing tools:

1. **Code analysis**

   Code analysis consists of different testing tools, including dynamic analysis and static analysis. You can apply different ones to tackle separate tests. For instance, some look for possible security flaws, while others check for usability. To run these tests, the developer will need to write code. Once this has been done, though, there's no human interference for the rest of the testing process.

# What are the types of automation testing?

2. **Unit tests**

Unit testing is all about checking individual components of the software or product, as you would developing ios technology for an iphone or android for samsung. This means that each element of the software is fully tested before the finished version is. These tests can be written by developers, but now that automation testing has come into play, there's no need.

3. **Integration tests**

Integration tests, also known as end-to-end tests, are often more complicated to set up than some other tests. The application models are integrated and tested as a group. This means that communication between each module can be tested, to figure out how well they work as a whole.

# What are the types of automation testing?

### 4. Automated accepted tests

Automated accepted tests (AAT) are similar to behavior-driven development (BDD) and automated acceptance test-driven development (AATDD). The acceptance test is created before a new feature is developed. It sets a precedent for the feature to meet and is usually written by developers, the business, and quality assurance (QA) in tandem. In future, they can also be used as regression tests.

### 5. Smoke tests

This type of testing is used to check whether the product is stable or not. If it's not stable, it gets sent back to the developers marked as an 'unstable build'. They can then run further tests if needed to identify the root cause of the problem. This diagram shows how the smoke test process works:

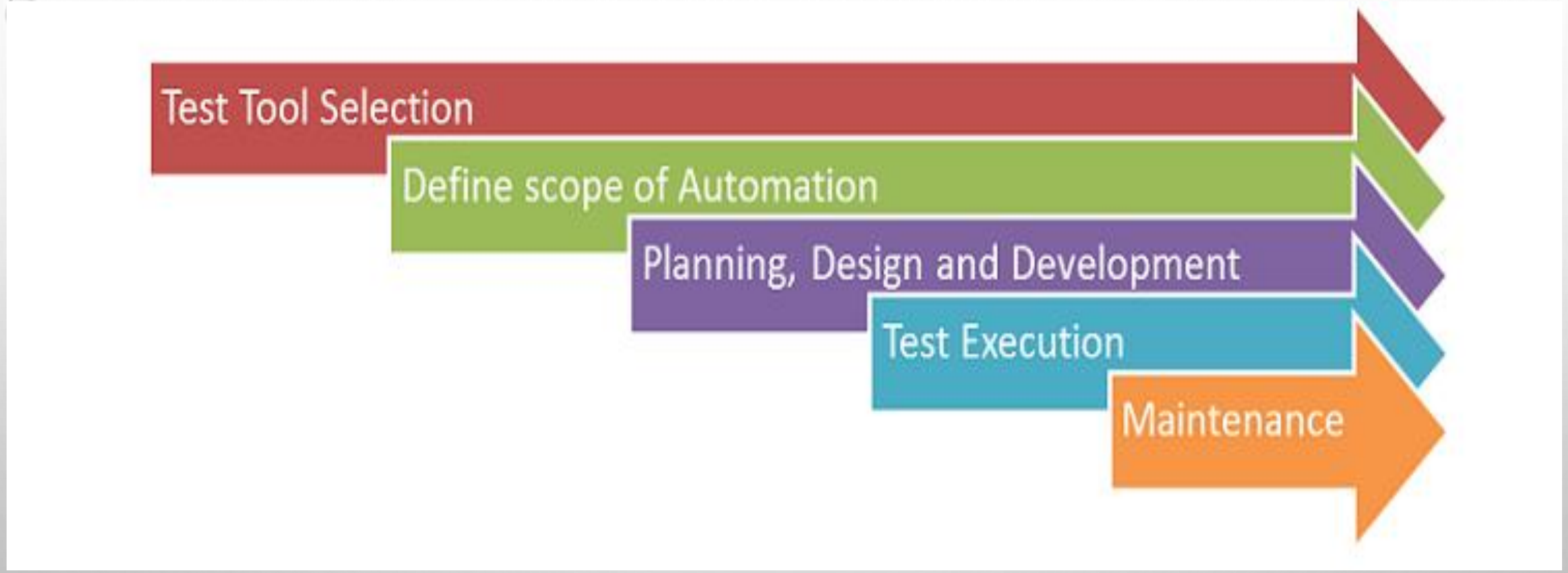
# Which Test Cases to Automate?

Test cases to be automated can be selected using the following criterion to increase the automation ROI

- High risk – business critical test cases
- Test cases that are repeatedly executed
- Test cases that are very tedious or difficult to perform manually
- Test cases which are time-consuming

- **TEST CASES ARE NOT SUITABLE FOR AUTOMATION:**
  - Test cases that are newly designed and not executed manually at least once
  - Test cases for which the requirements are frequently changing
  - Test cases which are executed on an ad-hoc basis.

# The automation test process?



# 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. **It's a good idea to conduct a proof of concept of tool on AUT.**
- We've been through the different types of automation testing. Now's the time to select which one best suits your needs. For instance, if your goal is to detect a specific bug within the software, you may be more inclined to use code analysis automation testing.
- You can select from a wide range of test automation tools and web apps on the market, such as selenium ide, webdriver, uft, ranorex, cucumber, testcomplete, and appium. You should be able to access some on microsoft with many offering a tutorial on how to use them. Some are even open source. So you'll need to have a solid understanding of each tool and how it could benefit your testing.

# 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
- Technical feasibility
- The extent to which business components are reused
- **The complexity** of test cases
- Ability to use the same test cases for cross-browser testing

# Planning, Design, and Development

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

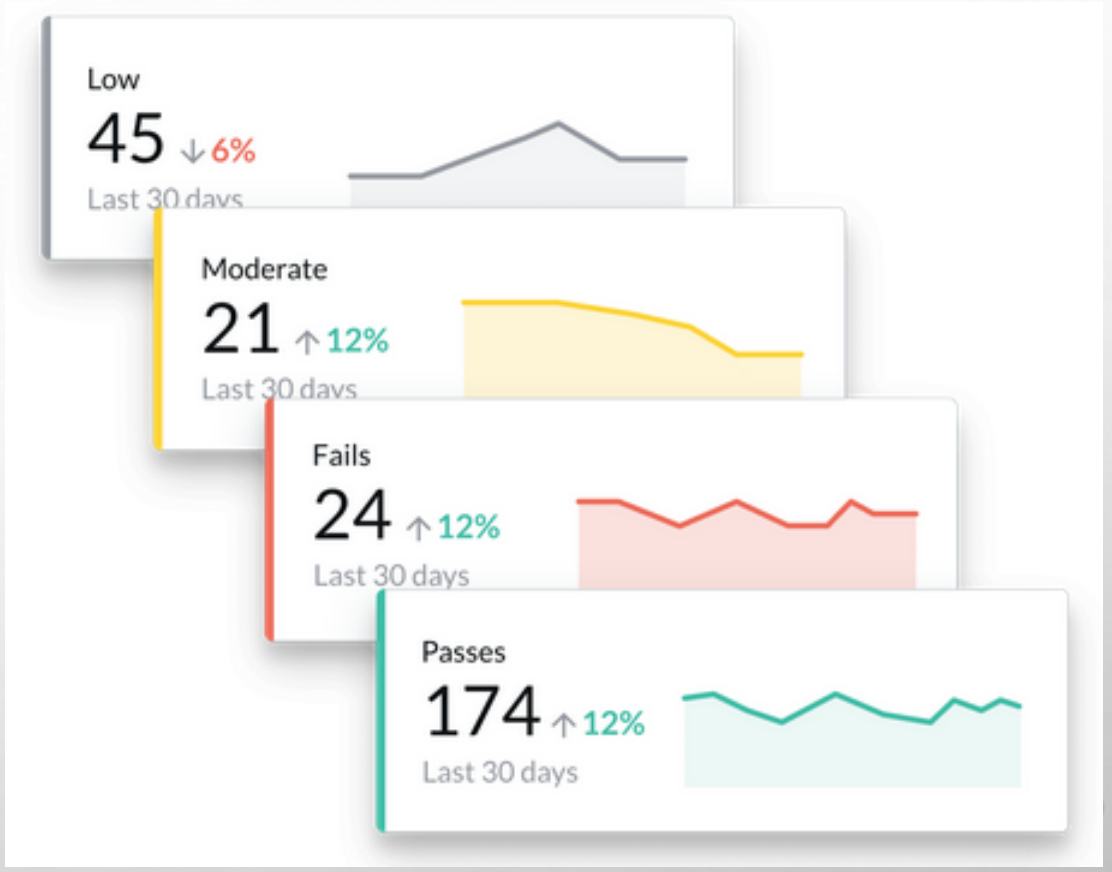
- Automation tools selected
- Framework design and its features
- In-scope and out-of-scope items of automation
- Automation testbed preparation
- Schedule and timeline of scripting and execution
- Deliverables of automation testing

# 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.
- Example: quality center is the test management tool which in turn it will invoke qtp for execution of automation scripts. Scripts can be executed in a single machine or a group of machines. The execution can be done during the night, to save time.

# 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.



# Framework for Automation

A framework is set of automation guidelines which help in

- Maintaining consistency of testing
- Improves test structuring
- Minimum usage of code
- Less maintenance of code
- Improve re-usability
- Non technical testers can be involved in code
- The training period of using the tool can be reduced
- Involves data wherever appropriate

# Test tools?

**There are four types of frameworks used in automation software testing:**

1.DATA DRIVEN AUTOMATION FRAMEWORK

2.KEYWORD DRIVEN AUTOMATION FRAMEWORK

3.MODULAR AUTOMATION FRAMEWORK

4.HYBRID AUTOMATION FRAMEWORK

# Popular automation tools

- 1. **Selenium** (https://www.softwaretestinghelp.com/selenium-tutorial-1/)
- 2.**QTP tutorials** (https://www.softwaretestinghelp.com/qtp-quicktest-professional-tutorial-1/)
- 3.**Soapui web services testing tool** *(https://www.softwaretestinghelp.com/web-services-api-testing-tool-soapui-tutorial-1/)*

4. **HP loadrunner for performance testing**

*(https://www.softwaretestinghelp.com/hp-loadrunner-load-testing-tool-training-tutorials/)*

# Popular automation tools

5. Sikuli GUI Automation Tool
6. PowerShell: Desktop Application UI Automation With PowerShell
7. Katalon Automation Recorder (Selenium IDE Alternative)
8. Geb Tool: Browser Automation Using Geb Tool
9. AutoIt: How to Handle Windows Pop-up Using AutoIt Cucumber: Automation Using Cucumber Tool and Selenium
10. Protractor Testing Tool for End-to-end Testing of AngularJS Applications

# Automation Frameworks

## I. Benefits of a test automation framework

Utilizing a framework for automated testing will increase a team's test speed and efficiency, improve test accuracy, and will reduce test maintenance costs as well as lower risks. They are essential to an efficient automated testing process for a few key reasons:

- **Improved test efficiency**
- **Lower maintenance costs**
- **Minimal manual intervention**
- **Maximum test coverage**
- **Reusability of code**

# Automation Frameworks

## II. Types of automated testing frameworks

1. Linear automation framework
2. Modular based testing framework
3. Library architecture testing framework
4. Data-driven framework
5. Keyword-driven framework
6. Hybrid testing framework

# Linear Automation Framework

With a linear test automation framework, also referred to as a record-and-playback framework, testers don't need to write code to create functions and the steps are written in a sequential order. In this process, the tester records each step such as navigation, user input, or checkpoints, and then plays the script back automatically to conduct the test.

**ADVANTAGES OF A LINEAR FRAMEWORK:**

- There is no need to write custom code, so expertise in test automation is not necessary.
- This is one of the fastest ways to generate test scripts since they can be easily recorded in a minimal amount of time.
- The test workflow is easier to understand for any party involved in testing since the scripts are laid out in a sequential manner.

# Linear Automation Framework

- This is also the easiest way to get up and running with automated testing, especially with a new tool. Most automated testing tools today will provide record-and-playback features, so you also won't need to plan extensively with this framework.

**DISADVANTAGES:**

- The scripts developed using this framework aren't reusable. The data is hardcoded into the test script, meaning the test cases cannot be re-run with multiple sets and will need to be modified if the data is altered.
- Maintenance is considered a hassle because any changes to the application will require a lot of rework. This model is not particularly scalable as the scope of testing expands.

# Modular Based Testing Framework

- Implementing a modular framework will require testers to divide the application under test into separate units, functions, or sections, each of which will be tested in isolation. After breaking down the application into individual modules, a test script is created for each part and then combined to build larger tests in a hierarchical fashion. These larger sets of tests will begin to represent various test cases.
- A key strategy in using the modular framework is to build an abstraction layer, so that any changes made in individual sections won't affect the overarching module.

# Modular Based Testing Framework

**ADVANTAGES OF A MODULAR FRAMEWORK:**

- If any changes are made to the application, only the module and it's associated individual test script will needs to be fixed, meaning you won't have to tinker with the rest of the application and can leave it untouched.
- Creating test cases takes less effort because test scripts for different modules can be reused.

**DISADVANTAGES OF A MODULAR FRAMEWORK:**

- Data is still hard-coded into the test script since the tests are executed separately, so you can't use multiple data sets.
- Programming knowledge is required to set up the framework.

# Library Architecture Testing Framework

- The library architecture framework for automated testing is based on the modular framework, but has some additional benefits. Instead of dividing the application under test into the various scripts that need to be run, similar tasks within the scripts are identified and later grouped by function, so the application is ultimately broken down by common objectives. These functions are kept in a library which can be called upon by the test scripts whenever needed.

# Library Architecture Testing Framework

**ADVANTAGES OF A LIBRARY ARCHITECTURE TESTING FRAMEWORK:**

- Similar to the modular framework, utilizing this architecture will lead to a high level of modularization, which makes test maintenance and scalability easier and more cost effective.
- This framework has a higher degree of reusability because there is a library of common functions that can be used by multiple test scripts.

**DISADVANTAGES:**

- Test data is still hard coded into the script. Therefore, any changes to the data will require changes to the scripts.
- Technical expertise is needed to write and analyze the common functions within the test scripts.
- Test scripts take more time to develop.

# Data-Driven Framework

- Using a data-driven framework separates the test data from script logic, meaning testers can store data externally. Very frequently, testers find themselves in a situation where they need to test the same feature or function of an application multiple times with different sets of data. In these instances, it's critical that the test data not be hard-coded in the script itself, which is what happens with a linear or modular-based testing framework.
- Setting up a data-driven test framework will allow the tester to store and pass the input/ output parameters to test scripts from an external data source, such as excel spreadsheets, text files, csv files, sql tables, or odbc repositories.
- The test scripts are connected to the external data source and told to read and populate the necessary data when needed.

# Data-Driven Framework

**ADVANTAGES OF A DATA-DRIVEN FRAMEWORK:**

- Tests can be executed with multiple data sets.
- Multiple scenarios can be tested quickly by varying the data, thereby reducing the number of scripts needed.
- Hard-coding data can be avoided so any changes to the test scripts do not affect the data being used and vice versa.
- You'll save time by executing more tests faster.

**DISADVANTAGES**

- You'll need a highly-experienced tester who is proficient in various programming languages to properly utilize this framework design. They will need to identify and format the external data sources and to write code (create functions) that connect the tests to those external data sources seamlessly.
- Setting up a data-driven framework takes a significant amount of time.

# Keyword-Driven Framework

- In a keyword-driven framework, each function of the application under test is laid out in a table with a series of instructions in consecutive order for each test that needs to be run. In a similar fashion to the data-driven framework, the test data and script logic are separated in a keyword-driven framework, but this approach takes it a step further.
- With this approach, keywords are also stored in an external data table (hence the name), making them independent from the automated testing tool being used to execute the tests. Keywords are the part of a script representing the various actions being performed to test the GUI of an application. These can be labeled as simply as ‘click,’ or ‘login,’ or with complex labels like ‘clicklink,’ or ‘verifylink.’
- In the table, keywords are stored in a step-by-step fashion with an associated object, or the part of the ui that the action is being performed on. For this approach to work properly, a shared object repository is needed to map the objects to their associated actions

# Keyword-Driven Framework

- **KEYWORD TABLE**

Once the table has been set up, all the testers have to do is write the code that will prompt the necessary action based on the keywords. When the test is run, the test data is read and pointed towards the corresponding keyword which then executes the relevant script.

| Step Number | Description | Keyword | Object |
|---|---|---|---|
| Step 1 | Click user portal link on homepage | clicklink | Login Button |
| Step 2 | Enter user name | Inputdata | Login Username |
| Step 3 | Enter password | Inputdata | Login password |
| Step 4 | Verify user log in information | verifylogin | |
| Step 5 | Log in to the application | login | Submit Button |

# Keyword-Driven Framework

**ADVANTAGES OF KEYWORD-DRIVEN FRAMEWORKS:**

- Minimal scripting knowledge is needed.
- A single keyword can be used across multiple test scripts, so the code is reusable.
- Test scripts can be built independent of the application under test.

**DISADVANTAGES:**

- The initial cost of setting up the framework is high. It is time-consuming and complex. The keywords need to be defined and the object repositories / libraries need to be set up.
- You need an employee with good test automation skills.
- Keywords can be a hassle to maintain when scaling a test operation. You will need to continue building out the repositories and keyword tables.

# Hybrid Test Automation Framework

- As with most testing processes today, automated testing frameworks have started to become integrated and overlap with one another. As the name suggests, a hybrid framework is a combination of any of the previously mentioned frameworks set up to leverage the advantages of some and mitigate the weaknesses of others.
- Every application is different, and so should the processes used to test them. As more teams move to an agile model, setting up a flexible framework for automated testing is crucial. A hybrid framework can be more easily adapted to get the best test results.