TEST AUTOMATION FRAMEWORKS: THE ULTIMATE GUIDE

Quality assurance (QA) is a major part of any software development. Software testing is the path to a bug-free, performance-oriented software application—one that also satisfies (or exceeds!) end-user requirements.

Of course, manual testing is quickly unscalable due to the rapid pace of development and ever-increasing requirements. Thus, a faster yet accurate testing solution was required, and automated testing became the ideal solution for this need. Automated testing does not mean replacing the entire manual testing process. Instead automated testing means:

1. Allowing users to automate most routine and repetitive test cases.
2. Freeing up valuable time and resources to focus on more intricate or complex test scenarios.

Introducing automated testing to a delivery pipeline can be a daunting process. Several factors—the programming language, user preferences, test cases, and the overall testing scope—directly decide what can and cannot be automated. However, if set up correctly, automated testing can be the backbone of the QA team to ensure a smooth and scalable testing experience.

Different types of automation frameworks came into prominence to aid in this endeavor. An automation framework allows users to easily set up an automated test environment that ultimately helps in providing a better ROI for both development and QA teams. In this article, we will have a look at different types of test automation frameworks available and their advantages and disadvantages.

(This article is part of our DevOps Guide. Use the right-hand menu to go deeper into individual practices and concepts.)

What is a test automation framework?

Before diving into different types of test automation frameworks, we need to understand what an automation framework is. Test automation is the process of automating repetitive and predictable testing scenarios.

A test automation framework is a set of guidelines or rules that can be used to define test cases. These test cases can then be configured and implemented using test automation tools such as Selenium, Puppeteer, etc., to the delivery process via a CI/CD pipeline.

A test automation framework will consist of practices and tools that are designed to create efficient test cases. These practices range from coding standards, test-data handling methods, object repository management, and managing access control to test environment and external tools, etc. However, testers have more freedom than this. Testers are:

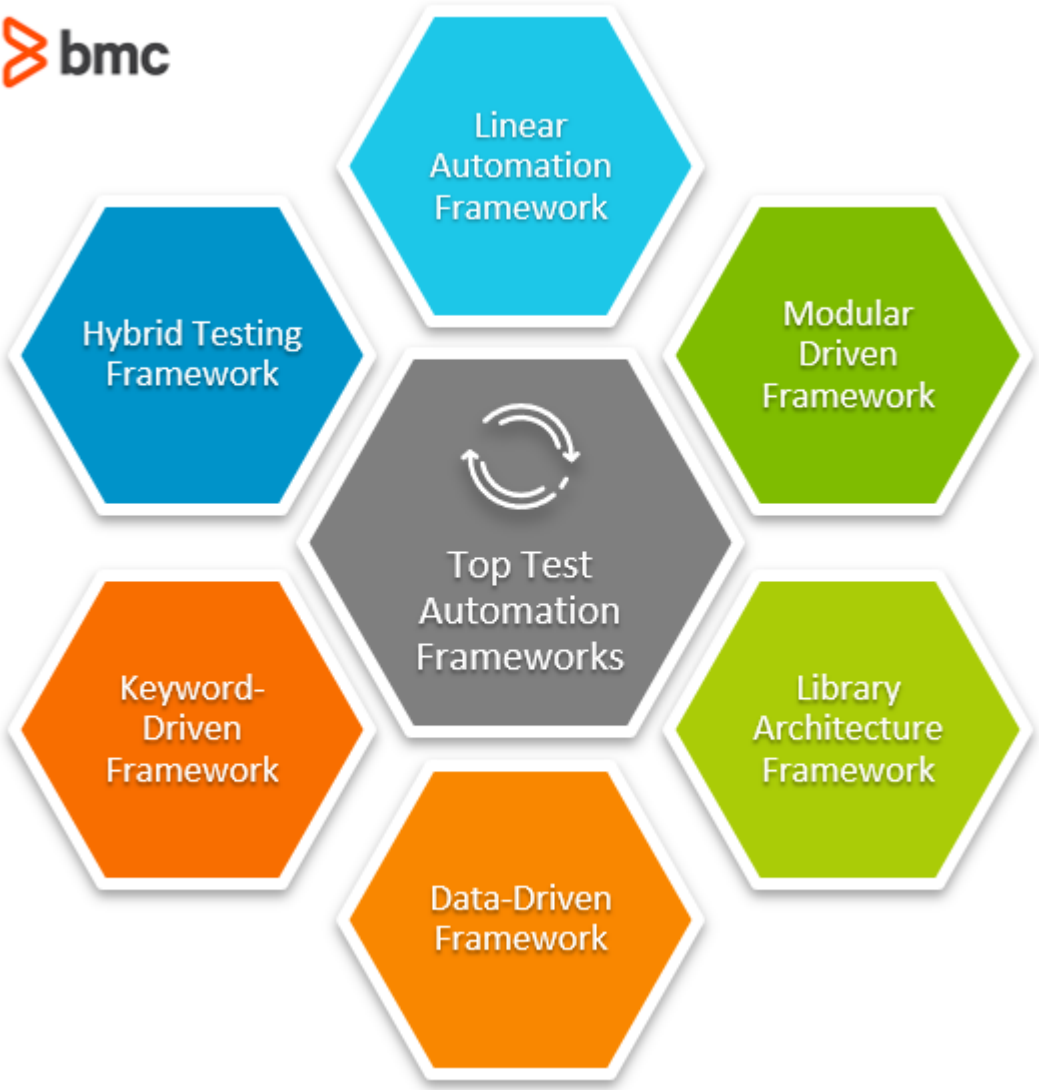
- Not confined to these rules or guidelines
- Free to create test cases in their preferred way

Still, a framework provides standardization across the testing process, leading to a more efficient, secure, and compliant testing process.

Advantages of a test automation framework

There are some key advantages of adhering to the rules and guidelines offered by a test automation framework. These advantages include:

- Increased speed and efficiency of the overall testing process
- Improved accuracy and repeatability of the test cases
- Lower maintenance requirements with standardized practices and processes
- Reduced manual intervention and human error
- Maximized test coverage across all areas of the application, from the GUI to internal application logic



Popular test automation frameworks

When it comes to test automation frameworks, there are six leading frameworks available these days. In this section, we will look at each of these six frameworks with regard to their architecture, advantages, and disadvantages:

- Linear automation framework
- Modular-driven framework
- Library architecture framework
- Data-driven framework
- Keyword-driven framework
- Hybrid testing framework

Linear Automation Framework

The linear framework or the record and playback framework is best suited for basic, introductory level testing.

In a linear automation framework, users target a specific program functionality, create test scripts in sequential order and run them individually. This process includes capturing all the tests like navigation, inputs, etc., and playing them back repeatedly to conduct the test.

Advantages of Linear Framework

- Does not require specific automation knowledge or custom code
- It is easier to understand test cases due to sequential order
- Faster approach to testing
- Simper implementation to existing workflows and most automation tools provides inbuilt tools for record and playback functionality

Disadvantages of Linear Framework

- Test cases are not reusable as they are targeted towards specific use cases or functions
- With static data, there is no option to run tests with different data sets as test data is hardcoded
- Maintenance can be complex as any change will require rebuilding test cases

Modular Driven Framework

This framework takes a modular approach to testing which breaks down tests into separate units, functions, or modules and will be tested in isolation. These separate test scripts can be combined to build larger tests covering the complete application or specific functionality.

(Learn about unit testing, function testing, and more.)

Advantages of Modular Framework

- Increased flexibility of test cases. Individual sections can be quickly edited and modified as tests are separated
- Increased reusability as individual test cases can be modified from different overarching modules to suit different needs
- The ability to scale up testing quickly and efficiently to include any new functionality

Disadvantages of Modular Framework

- Can be complex to implement and require proper programming knowledge to build and set up test cases
- Cannot be used with different test data sets in a single test case

Library Architecture Framework

This framework is derived from the modular framework that aims to provide a greater level of modularity to testing by breaking down tests by units, functions, etc.

The library architecture framework identifies similar tasks within test scripts and groups them by function. These modular parts aren't directly about function—they're more focused on common objectives. Then these functions are stored in a library sorted by their objectives, and test scripts call upon this library to obtain different functionality when testing.

Advantages of Library Architecture Framework

- A high level of modularity leads to increased scalability of test cases
- Increased reusability as libraries can be used across different test scripts
- Can be a cost-effective solution due to its reusability, especially in larger projects

Disadvantages of Library Architecture Framework

- Can be complex to set up and integrate into delivery pipelines
- Technical expertise is required to identify and modularize the common tasks
- Test data are static as they are hardcoded in script with any changes requiring direct changes to the scripts

Data-Driven Framework

The main feature of the data-driven framework is that it decouples data from the script logic. It is the ideal framework when users need to test a function or scenario with different data sets but still use the same internal logic.

In data-driven frameworks, values such as inputs and outputs are passed as parameters to test scripts from external data sources such as variable files, databases, etc.

Advantages of Data-Driven Framework

- Decoupled approach to data and logic leads to increased reusability of test cases while providing the ability to test under different data sets without modifying the test case
- Handle multiple scenarios using the same test scripts with varying sets of data, which leads to faster test environments
- Since there is no need to hardcode data, scripts can be changed without affecting the overall functionality
- Easily adaptable to suit any testing need

Disadvantages of Data-Driven Framework

- One of the most complex frameworks to implement as decoupling data and logic will require expert knowledge both in automation and the application itself
- Can be time-consuming and a resource-intensive process to implement in the delivery pipeline

Keyword-Driven Framework

The keyword-driven framework takes the decoupling of data and the logic introduced in the data-driven framework a step further. In addition to the data being stored externally, specific keywords that are associated with different actions and used to test the GUI are also stored externally to be referenced at the test execution.

It makes keywords independent entities that reference specific functions or actions that are associated with specific objects. Users write code to prompt the necessary keyword-based action,

and the appropriate script is executed within the test when the keyword is referenced.

Advantages of Keyword-Driven Framework

- Test scripts can be built independently of the application
- Increased reusability and flexibility while providing a detailed approach to categorize test functionality
- Reduced maintenance requirements compared to non-decoupled frameworks

Disadvantages of Keyword-Driven Framework

- One of the most complex frameworks to configure and implement, requiring a considerable investment of resources
- Keywords need to be scaled according to the application testing needs, which can lead to increased complexity with each test scope or requirement change

Hybrid Testing Framework

A hybrid testing framework is not a predefined framework with its architecture or rules but a combination of previously mentioned frameworks.

Depending on a single framework will not be a feasible endeavor with the ever-increasing need to cater to different test scenarios. Therefore, different types of frameworks are combined in most development environments to best suit the application testing needs while leveraging the strengths of each framework and mitigating the disadvantages.

With the popularity of DevOps and agile practices, more flexible frameworks are needed to cope with the changing environments. Therefore, a hybrid approach provides the best solution by allowing users to mix and match frameworks to obtain the best results for their specific testing requirements.

Customizing your frameworks

Selecting a test automation framework is the first step towards creating an automated testing environment. However, relying on a single framework has become a near-impossible task due to the ever-evolving nature of the technological landscape and rapid development cycles. That's why the hybrid testing framework has gained popularity—for enabling users to combine different test automation frameworks to build an ideal automation framework for their needs.

Even if you are new to the automation world, you can start with a framework with many built-in solutions, build on top of it and customize it to create the ideal framework.

Related reading

- BMC DevOps Blog
- Testing in DevOps: Concepts, Best Practices & More
- Shift Left Testing: What, Why & How To Shift Left
- Performance Testing, Load Testing & Stress Testing Explained
- What Is Continuous Testing?

- What's Testing as a Service? TaaS Explained