What is a Test Framework?

Before diving into the most common types of frameworks and their benefits, let’s clarify what a test automation framework actually is**. A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently.**

These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

While these are not mandatory rules and testers can still script or record tests without following them, using an organized framework typically provides additional benefits that would otherwise be missed out on.

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**

A common trend to minimize risk is to test earlier in the Test Automation Framework. By using tools such as [TestLeft](https://smartbear.com/product/testleft/overview/), you can test in your own IDE. Download a free [30-day trial](https://smartbear.com/product/testleft/free-trial/)to Automate in your IDE today!

Types of Automated Testing Frameworks

There are six common types of test automation frameworks, each with their own architecture and differing benefits and disadvantages. When building out a test plan, it’s important to [choose the framework that is right for you.](https://smartbear.com/learn/automated-testing/choosing-a-test-framework/)

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.
* 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.

**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.

**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.

**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](https://smartbear.com/learn/automated-testing/why-gui-testing/) 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 Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step Number** | **Description** | **Keyword** | **Object** | **Action** |
| 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 |  |

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.

**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.

Conclusion

One recommended approach for implementing a hybrid framework for automated testing, is to find a tool that can quickly and easily adapt to your processes. [When choosing an automated testing tool,](https://smartbear.com/learn/automated-testing/selecting-automated-testing-tools/) you should look for one that is flexible and can support a wide range of applications and languages. This will enable your team, regardless of background and skill set, to contribute to your testing efforts