**Frameworks**

A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

The advantages can be in different forms like ease of scripting, scalability, modularity, understandability, process definition, re-usability, cost, maintenance etc. Thus, to be able to grab these benefits, developers are advised to use one or more of the Test Automation Framework.

**Advantage of Test Automation framework**

1. Reusability of code
2. Maximum coverage
3. Recovery scenario
4. Low cost maintenance
5. Minimal manual intervention
6. Easy Reporting

**The most popularly used Test Automation Frameworks:**

1. Module Based Testing Framework
2. Library Architecture Testing Framework
3. Data Driven Testing Framework
4. Keyword Driven Testing Framework
5. Hybrid Testing Framework
6. Behavior Driven Development Framework

**#1) Module Based Testing Framework**

Module based Testing Framework is based on one of the popularly known OOPs concept – Abstraction. The framework divides the entire “Application Under Test” into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one modules.

**Pros:**

* Modular division of scripts leads to easier maintenance and also the scalability of the automated test Scripts are independent to write.

**Cons:**

* The main problem with modular frameworks is that the test script have test data embedded in them, which will become problem when updating the code /script.
* It is also difficult whenever a test step fails which has to be find out by debugging where the test case failed.

**#2) Library Architecture Testing Framework**

The Library Architecture Testing Framework is fundamentally and foundationally built on Module Based Testing Framework with some additional advantages. Instead of dividing the application under test into test scripts, we segregate the application into functions or rather common functions can be used by the other parts of the application as well. Thus we create a common library constituting of common functions for the application under test. Therefore, these libraries can be called within the test scripts whenever required.

The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

**Pros:**

1. Like Module Based Framework, this framework also introduces high level of modularization which leads to easier and cost efficient maintenance and scalability too.
2. As we create common functions that can be efficiently used by the various test scripts across the Framework. Thus, the framework introduces a great degree of re-usability.

**Cons:**

1. Like Module Based Framework, the test data is lodged into the test scripts, thus any change in the test data would require changes in the test script as well.
2. With the introduction of libraries, the framework becomes a little complicated.

### ****#3) Data Driven Testing Framework****

While automating or testing any application, at times it may be required to test the same functionality multiple times with the different set of input data. Thus, in such cases, we can’t let the test data embedded in the test script. Hence it is advised to retain test data into some external data base outside the test scripts.

Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The external databases can be property files, xml files, excel files, text files, CSV files, ODBC repositories etc. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.

**Pros:**

1. The most important feature of this framework is that it considerably reduces the total number of scripts required to cover all the possible combinations of test scenarios. Thus lesser amount of code is required to test a complete set of scenarios.
2. Any change in the test data matrix would not hamper the test script code.
3. Increases flexibility and maintainability
4. A single test scenario can be executed altering the test data values.

**Cons:**

1. The process is complex and requires an extra effort to come up with the test data sources and reading mechanisms.
2. Requires proficiency in a programming language that is being used to develop test scripts.

### ****#4) Keyword Driven Testing Framework****

The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.

These set of code are known as Keywords and hence the framework is so named. Key words are self-guiding as to what actions needs to be performed on the application.

The keywords and the test data are stored in a tabular like structure and thus it is also popularly regarded as Table driven Framework. Take a notice that keywords and test data are entities independent of the automation tool being used.

**Pros:**

1. In addition to advantages provided by Data Driven testing, Keyword driven framework doesn’t require the user to possess scripting knowledge unlike Data Driven Testing.
2. A single keyword can be used across multiple test scripts.

**Cons:**

1. The user should be well versed with the Keyword creation mechanism to be able to efficiently leverage the benefits provided by the framework.
2. The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

### ****#5) Hybrid Testing Framework****

As the name suggests, the Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.

### ****#6) Behavior Driven Development Framework****

Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc. Such frameworks do not necessarily require the user to be acquainted with programming language. There are different tools available for BDD like cucumber, Jbehave etc.

Though the above pictorial representation of a framework is self-explanatory but we would still highlight a few points.

1. **Object Repository**: Object Repository acronym as OR is constituted of the set of locators types associated with web elements.
2. **Test Data:** The input data with which the scenario would be tested and it can be the expected values with which the actual results would be compared.
3. **Configuration File/Constants/ Environment Settings**: The file stores the information regarding the application URL, browser specific information etc. It is generally the information that remains static throughout the framework.
4. **Generics/ Program logics/ Readers**: These are the classes that store the functions which can be commonly used across the entire framework.
5. **Build tools and Continuous Integration**: These are the tools that aids to the frameworks capabilities to generate test reports, email notifications and logging information.

**Page Object model:**

A better approach to script maintenance is to create a separate class file which would find web elements, fill them or verify them. This class can be reused in all the scripts using that element. In future if there is change in the web element, we need to make change in just 1 class file and not 10 different scripts.

This approach is called Page Object Model (POM). It helps make code more readable, maintainable, and reusable.

* Page Object Model is a design pattern to create Object Repository for web UI elements.
* Under this model, for each web page in the application there should be corresponding page class.
* This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
* Name of these methods should be given as per the task they are performing i.e., if a loader is waiting for payment gateway to be appear, POM method name can be waitForPaymentScreenDisplay().

**Advantages:**

* **Object Repository**: You can create an Object Repository of the fields segmented page-wise. This as a result provides a Page Repository of the application as well. Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.
* **Functional Encapsulation**: All possible functionality or operations that can be performed on a page can be defined and contained within the same class created for each page. This allows for clear definition and scope of each page's functionality.
* **Low maintenance**: Any User Interface changes can swiftly be implemented into the interface as well as class.
* **Programmer Friendly**: Robust and more readable. The Object-oriented approach makes the framework programmer friendly.
* **Low Redundancy**: Helps reduce duplication of code. If the architecture is correctly and sufficiently defined, the POM gets more done in less code.
* **Efficient & Scalable**: Faster than other keyword-driven/data-driven approaches where Excel sheets are to be read/written.

**Disadvantages**

* **High Setup Time & Effort**: Initial effort investment in development of Automation Framework is high. This is the biggest weight of POM in case of web applications with hundreds/thousands of pages. It is highly suggested that if this model is decided to be implemented, then it should be done parallel to development of the application. Refer [V-Model](http://www.tutorialspoint.com/sdlc/sdlc_v_model.htm) for Software Development Life Cycle.
* **Skilled labor**: Testers not technically sound or aware of programming best practices are a nightmare in this case. Perhaps this is the biggest mistake to make, employing unskilled labor in hopes of training them during implementation. Unskilled testers need to undergo a Training Boot Camp to be ready for such an undertaking. Also the Architecture of the framework should be defined clearly and completely before development in order to avoid any loopholes in later stages. Every application is different and it may require the automation framework to be significantly tailored towards it.
* **Specific**: Not a generic model. Automation Framework developed using POM approach is specific to the application. Unlike keyword-driven/data-driven frameworks, it is not a generic framework.