**Test Automation framework**

Test automation framework to be explained in laymen terms as the set of guidelines such as coding standards, data handling, report presentation and bringing the repository system into place. With the following of the guidelines we can make the suite more re-usable with as less maintenance as possible

**What is framework?**

Framework is all about defining set protocols, rules and regulations to make the task more standardized.

**Why framework?**

The framework standard makes us to be more procedural oriented. This standard practice will help everyone to understand the way to achieve a specific task and we have a great control of tracking things and ultimately we end up have hassle free maintenance.

In software development life cycle we rely on two we follow two major approaches in order to build a successful software product let’s see what are they and role of framework

Let’s quickly discuss the two common framework approaches that are followed most commonly in development and testing streams. These are rule based approaches for the successful implementation of the task.

**Test driven development approach (TDD)**

This is one of the best agile development methodologies, in this approach we write tests first then the goal is to make sure the tests go through. This way we can get rid of redundant work and build very accurate system.

Developers will establish a framework that all unit tests are automated. The code is then added to make sure the tests are passed and then the next implementation takes place. this is the repeated process that takes place until the product is completely built.

Behavior driven development approach

This is also a test first approach but these acceptance tests were written by either business analysts or testers. Unlike the TDD, this driven approach will try to certify the implemented functionality. The framework in this approach is built in a fashion that it validates the full functionalities then produces a meaningful report to understand the results.

**Role of framework in test automation**

* Easy to maintain the test suite
* Easy to transfer knowledge to any new joiner to the team
* Code re-usability

**Components in test automation framework**

Thus far we have discussed about the framework in a more general way, let’s take a moment to correlate that to the test automation. What are those rules and components that are collectively called as framework in a test suite?

**Dependency management**

Whilst creating a project it’s essential to rely on a dependency management and build tools to ensure all in the team uses same version of a particular library. We should never consider loading the libraries from the build path. Omission of this rule can lead to a great confusion and we might observe things that are working in one machine, may fairly get failed on others end. Also if there are version mismatches due to hardcoding the libraries by every individual we can’t root cause the problems.

Maven, Gradle can help us on resolving the dependencies also to build the code

**Reporting results**

Results are mandatory to be shared after test execution, so integrating a decent reporting API will help us publishing a report that can be deciphered by all the stakeholders. Report creation must be taken utmost care as it shows how many tests are passed and failed. The failed ones can later be analyzed to see if they are true defects are any script or environment issues.

Best reports that are used are extent reports, allure reports and testing html reports

**Test Runner**

In a sophisticated test suite, we will have hundreds to thousand numbers of test cases. Without having a proper test runner tool we can manage the test execution. These tools will help us running the tests with different set options, some are a below

* Prioritize the tests
* Making one test dependent on other
* Execution the same test for the number of times we want
* Running the failed test cases
* Grouping the test cases and run them in parallel of sequential fashion

Best examples of test runner tools are J-unit, N-unit, testNG, jasmine..etc

**Version controlling system**

Integration of VCS is essential as we as a team commit lot of changes to the code base on a daily basis. It’s highly difficult to integrate the changes. Proper education on pulling he code as first activity before we start our day then pushing at the end of the day after a review will help us maintaining the code base up to the date.

Best examples of VCS systems are GIT, BitBucket, SVN..etc

**Logging mechanism**

Logs are anywhere in the world are treated to be important as that helps greatly in triaging and root cause a problem that’s occurred. We need to write log files to a file for the each step we perform during test execution. This log acts as evidence to the malfunction we observe.

Log4j is the one of the best log tool to be integrated

**Third party tool integration**

At times in order to perform specific tasks we might not be able to achieve with the existing framework then we should consider adding any third party tool that can help us doing that, this is how we are making our framework more sophisticated and precise

Example for the third party tools are VNC viewer, AutoIT, winnium…etc. we will need these tools while there is a need to perform some windows actions in selenium framework

**Screenshots capturing**

We should have a disciplined screenshot capturing mechanism and storing them in a repository to help with understand the issue that occurred during failures. We can take a call as whether to capture only on failure cases or on any test step depending upon the memory we have.

**Types of test automation frameworks**

Given the design implementation and other properties we have various types of test automation frameworks and they are as below

**Linear framework**

Linear framework is implemented in such a way that the test is directly written in a file as a straight forward test. This test can also be designed by using any record and play back tools. This kind of approach however is simpler to design it leaves us doing lot of maintenance.

**Advantages**

Easy to develop scripts

Not much coding experience is required, only tool experience would do the trick

**Disadvantages**

Since the test is written directly, it’s very difficult to maintain the script when there are changes to the applications.

**Modular framework**

Modular framework is designed in such a way that, tests of common modules were first identified and then design those as reusable or generic methods. Since the common methods were designed as wrapper methods we can very well re-use them in any other test cases.

This kind of approach is followed when the application development is being done as individual micro services. Tests are developed for a specific service and only those can be invoked on demand.

**Advantages**

Since the common flows have been documented as reusable methods, re-usability increases

Less maintenance required

**Disadvantages**

Still the data is hard coded in tests

**Data driven framework**

Data driven framework has been identified as one of the powerful frameworks across all. In this type we can service the test data that’s needed via excel, jdbc sources, csv files. The tests become more generic as the data has been separated from the test code. With the help of various data script flow can be altered and that way we can achieve more test coverage.

In this we have classified the data as two types as

**Parameterization**

Any data that looks to be static and we don’t see dynamic changes can be considered as parameter. The best examples for this kind of data are urls, jdbc urls, endpoint details and environment details. We tend to read this data from the properties file or from xml file

**Data provider**

This is the real test data needed for the test case. We basically read this data from the external sources such as excel workbooks, CSV files, JDBC sources. This data has the ability to alter the flow of the test case. This facilitates us to provide more coverage with minimum code base.

**Advantages**

Since the data is separated from the tests, tests become more versatile

Data can easily be created as per the requirement

Easy to maintain the scripts

**Key driven framework**

In keyword driven framework, the code base / test case is driven by the action based keyword. In this framework we have all the actions identified and list as keywords then map to the application functionality. The keywords again are consumed from an external excel workbook or from a separate java class. This framework can be even developed without application by assuming the functional flow but it needs high expertise and it can lead to a re-work too if things don’t go right.

**Advantages**

Since the code base is developed as per actions, methods developed for a particular action can be re-used.

**Hybrid framework**

Hybrid framework is a customized one, all the best things from the different frameworks are pursued in this. When we say customization, there is no standard practice as such, it’s all about the requirement within a particular project. We might see mix of data driven, keyword and modular practices to make the framework more versatile and flexible to maintain.

**Behavior driven development framework**

Behavior driven development approach is also known as ATDD (acceptance test driven development). In this model the tests are driven by a test steps written by ether BA or tester in plain English that is understood by everybody. A spec file or feature file is created for a specific test case in gherkin language then the code base is developed accordingly.

**How is this model helping?**

Ever since the agile model of development has emerged, we started giving more priority to parallel testing and early testing alongside development. There were some challenges in understanding the test case developed in the above discussed other frameworks. Tests appear to be more technical so that it’s getting difficult to non-technical stakeholders to understand. In this approach as the test case written in a specific language which is gherkin (more or less English). Given the design its easy for everyone to say what’s happening with the test also report makes more sense too.

Examples of this kind of frameworks are- cucumber, gauge..etc

**Conclusion**

There is no wrong in being a process nerd, because unless we have protocols defined even in our lives we can’t be sure of the progress and get the things right on the very first opportunity. In the same fashion with the help of above defined rules we can achieve the tasks hence thanks to the framework and its defined best practices. We should still be under an impression that the best yet to come, given that sense we expect lot more enhancements to make our lives far easier at work. Change is inevitable be ready to adopt.

Thanks for walking through this long, I reckon you would have had some learning. ☺