**Name: A. Shiva Surya Saran**
**Roll No.: BE-B 20**

# Software Testing and Quality Assurance

## Assignment 3

## Unit 3: Software Test Automation

### *1. What is test automation? Scope of automation.*

Automation Testing or Test Automation is a software testing technique that performs using special automated testing software tools to execute a test case suite.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

The scope of automation is the area of 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

## *2. List and explain skills needed for automation.*

Here are the must-have skills for an automation tester:

i. **Focus on analytical thinking**
An instinct for analytics and logical application of concepts is important. Once the business team provides the business requirement document, the automation testing team should focus on understanding every aspect of the feature very well from an automation perspective.

ii. **Understanding of programming languages**
he automated test tools use programming languages like Java, Python, Perl, Vb script, etc. The automation tester needs to be proficient in these programming languages. The thought process of the automation tester should be to identify and cover all the possible modules that demand automation. The automation tester must have good coding skills in order to design the test scripts.

iii. **Good functional testing skills**
To excel as an automation tester, the tester should have sound knowledge and experience of functional testing performed manually.

iv. **Expertise in creation of test scripts**
Automation testers should have programming knowledge if the organizations are dependent on automated tools like UFT (Unified functional tester) or QTP (Quick Test Professional).

v. **Possess good knowledge on Automated testing tools**
The automation testers should be well-equipped with ample knowledge about the automated testing tools present in the market that eventually help to optimize the overall testing process.

vi. **Clear understanding of business requirements**
An automation tester must be familiar with:
- The programming language on which the application is developed.
- Browser or device requirement where the application is to be accessed by the end-users.
- APIs or any web services connected to the application and their working.
- Which databases are used for storing the backend information?
- Working of all the modules and features in the application.
- Is there a need for manual testing for testing some areas in the application?
- Planned test execution time by the automation tester.
- If there are any defects that were deferred in the last release and expected to be fixed in this release and the overall impact of this fix on the application.
- Test execution completion delivery date.
- Project release timelines.

vii. **Well versed with agile, DevOps and continuous delivery**
As agile methodology involves frequent changes, it is essential to have an

automation testing process in place for the same. Automation testers can automate the test scripts for a module to be able to respond to frequent customer induced requirement changes.

viii. **Maintain good communication and interaction with stakeholders**

ix. **Curious to learn new technologies and trends**

x. **Able to assess and mitigate the risk**

xi. **Good problem-solving and reporting skills.**

## *3. What are some of the challenges in automating the testing of GUI portions of an application? How do these compare with the automation of back- end testing?*

The major challenges in automating testing of GUI portions of an application are as follows:

1. Continuously changing UI
   Considering the latest technologies emerging these days, our web applications need to be upgraded. This calls for a change. This change can be related to any new version, integration with the third party tools or maybe sometimes, there are new functionalities that we want to implement in our web apps. Obviously, we might miss out to develop our UI tests in a way to utilize later.

2. Increasing Complexity of Testing Web Elements
   Latest web functionalities that we implement in our web applications can include various web elements. Those elements can be embedded frames and other products as well. Sometimes, large enterprise websites contain complex flowcharts, diagrams, maps, etc. These make the website's UI test automation complex.

3. Handling Multiple Errors
   Error handling has been an issue with UI automation testing. Whenever there are complex UI test scenarios with tight deadlines, most time is utilized in creating UI test scripts. Thus, testers choose manual testing over automation for UI testing. Having said that, error handling becomes extremely difficult when you manually revoke the error messages and automate the same.

4. Choosing a Random Automated UI Testing Tool
   There are numerous test automation tools available in the industry. In fact, there are certain tools that showcase themselves specialized in UI automation testing. And, when you try them, you will end up wasting your time and efforts. But, maybe due to budget constraints or project complexities, the enterprises end up choosing the inappropriate tools.

5. Maintaining Automated UI Scripts
   What happens in UI automated test scripts is, that it is difficult to maintain. Web developers often make changes to UI rather to the logic of the features and functionalities. With this, the UI test scripts fail each time there are new changes to the UI. Hence, maintenance of the UI

scripts has been a challenge for long.

6. Creating Effective UI automation tests take time
Everybody is talking about automating tests at DevOps speed. But, this does not affect creating UI automated tests as it takes hell lot of time and patience. Also, you know that there are a few tools that do not support recording UI automation tests. This creates challenges for developing automated test scripts for UI, effective.

7. Calculating ROI for UI automation
I keeps on changing, the tests will change accordingly. It will double the time that UI automation testing will take, delaying the process of delivery. Though test automation always comes with justified ROI attached.

8. Reviewing Quality Code standards
Each developer involved in developing an application will have different coding and code commenting styles. This makes it difficult for other developers as well as testers to review it sooner or later for modifications or maintenance.

Comparison with the automation of Backend testing:

- Frontend Testing checks the presentation layer of a 3 Tier Architecture whereas backend testing checks the application and database layer of a 3 Tier Architecture.
- It is always performed on the GUI whereas backend Testing involves databases and business logic testing.
- It does not need any information to be stored in a database, but backend testing needs information stored in the database.
- It is essential to check the overall functionality of the application while backend testing is important to check for deadlock, data corruption, data loss, etc.

## *4. What are the selection criteria of automated testing tool?*

Success in any test automation depends on identifying the right tool for automation. Selecting the “correct” Testing Tool for your project is one of the best ways to achieve the project target. To select the most suitable testing tool for the project, the Test Manager should follow the below tools selection process.

i. Identify the requirement for tools
Precisely identify your test tool requirements. All the requirement must be documented and reviewed by project teams and the management board.

ii. Evaluate the tools and vendors
After baselining the requirement of the tool, the Test Manager should, Analyze the commercial and open source tools that are available in the market, based on the project requirement, shortlist which meets the criteria and Evaluate the quality of the tool by taking the trial usage & launching a pilot.

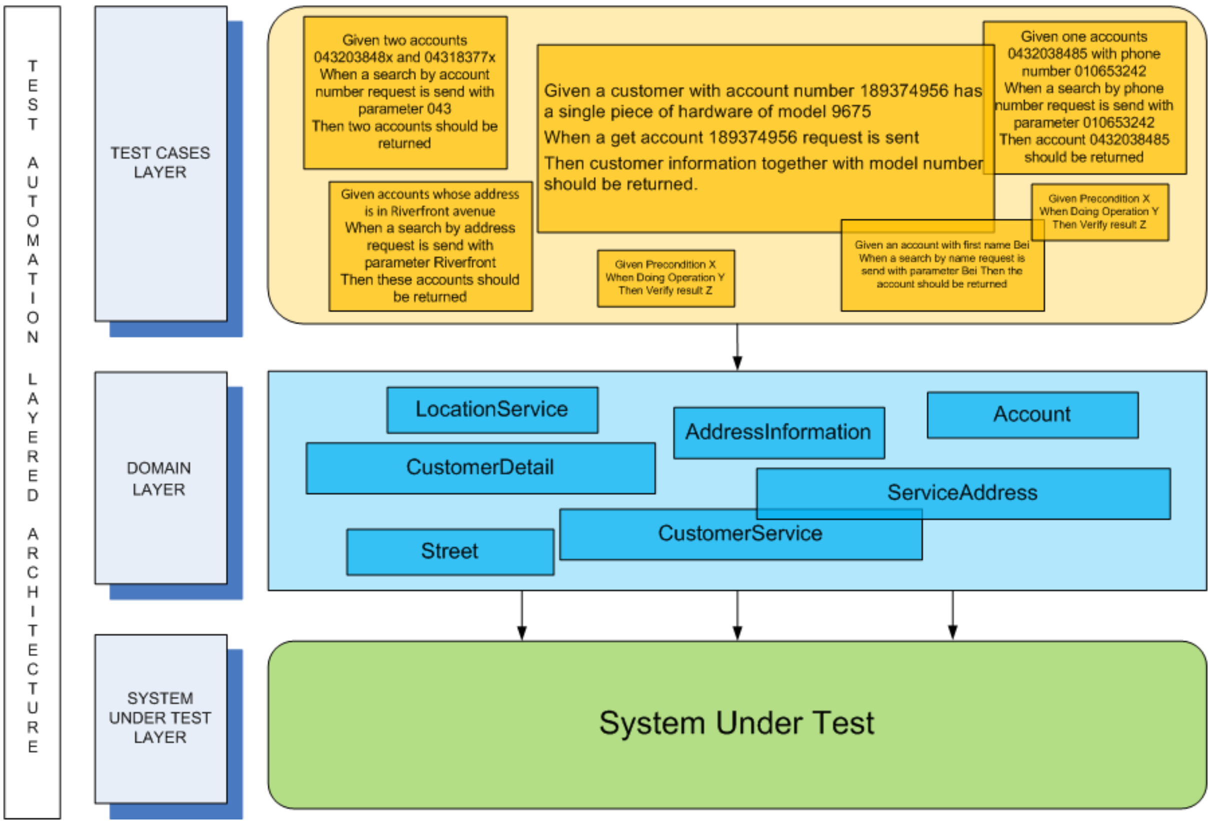
iii. Estimate cost and benefit
A cost-benefit analysis should be performed before acquiring or building a tool.

iv. Make the final decision
To make the final decision, the Test Manager must have a strong awareness of the tool. It means you must understand which is the strong points and the weak points of the tool and Balance cost and benefit

## 5. *Design an architecture for automation*

In layered architecture of test automation code is divided into three layers:

i. Test cases, focusing on the test logic of the application.
ii. The domain layer, modelling the system under test in domain terms, encapsulating http requests, browser control, result parsing logic and providing an interface for the test cases layer.
iii. The system under test, which layer 2 will operate directly on.


TEST AUTOMATION LAYERED ARCHITECTURE
TEST CASES LAYER
DOMAIN LAYER
SYSTEM UNDER TEST LAYER
Given two accounts 043203848x and 04318377x When a search by account number request is send with parameter 043 Then two accounts should be returned
Given a customer with account number 189374956 has a single piece of hardware of model 9675 When a get account 189374956 request is sent Then customer information together with model number should be returned.
Given one accounts 0432038485 with phone number 010653242 When a search by phone number request is send with parameter 010653242 Then account 0432038485 should be returned
Given accounts whose address is in Riverfront avenue When a search by address request is send with parameter Riverfront Then these accounts should be returned
Given Precondition X When Doing Operation Y Then Verify result Z
Given an account with first name Bei When a search by name request is send with parameter Bei Then the account should be returned
Given Precondition X When Doing Operation Y Then Verify result Z
LocationService
AddressInformation
Account
CustomerDetail
ServiceAddress
CustomerService
Street
System Under Test

***Layered Architecture of Test Automation***

### *Test Cases Layer*

All (and only) test logic resides here. Test logic can be expressed concisely, with the help of the layer below. Test cases for different stories, scenarios, and corner cases rely on the same piece of code in the layer below, the only difference is in parameters or test data representing different cases.

***Domain Layer***

This layer will encapsulate operations to the system under test, like url concatenation, response xml/html parsing, rich-client GUI/browser control, etc. It will present the system under test in domain language, rather than in terms of xpath, sql, or html.

***System Under Test layer***

It's the system under which it's being tested for.

## *6. Short note on Jmeter and JUnit*

### Jmeter

The Apache JMeter is pure Java open source software, which was first developed by Stefano Mazzocchi of the Apache Software Foundation, designed to load test functional behaviour and measure performance.
JMeter is used to analyse and measure the performance of web application or a variety of services.

### Junit

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

## *7. List generic requirement for test tools*

- Data driven capabilities
- Debugging and logging capabilities
- Platform independence
- Extensibility & Customizability
- E-mail Notifications
- Version control friendly
- Support unattended test runs

## 8. *What are the difference in testing automation in agile and waterfall model.*

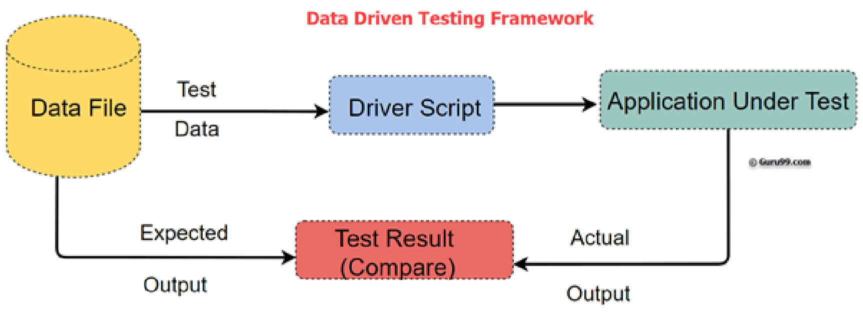
Automation Testing is normally feasible when the application is stable, steady and the requirement is involving with a real considerable amount of time and in most cases involving a set of very skilful automation expert resources as well as a considerable amount of set-up costs. The basic purpose of Automation Testing is to reduce costs over a long time and to ensure no new defects have been introduced because of existing test cases.

Automation testing by the very nature of the technology is not exploratory in nature since the main role of Automation Testing is saving time and reducing costs. Automation Testing is not meant to come up with new and innovative defects. Automation Testing aims at mostly confirmation of the already existing.

## 9. *What is data Driven testing (DDT)? Explain data driven testing framework.*

Data Driven Testing is a software testing method in which test data is stored in table or spreadsheet format. Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table. It is also called table-driven testing or parameterized testing.

In DDT, Data Driven Framework is used as an automation testing framework in which input values are read from data files and stored into variables in test scripts. It enables testers to build both positive and negative test cases into a single test. Input data in data driven framework can be stored in single or multiple data sources like .xls, .xml, .csv and databases.



## *10. What challenges faced at the time of automation.*

Here are the most common test automation challenges:

i. Testing the complete application
It's not possible to test each and every combination both in the Manual as well as in Automation Testing.

ii. Misunderstanding of company processes
There are some myths in testers that they should only go with company processes even these processes are not applicable for their current testing scenario. This results in incomplete and inappropriate Application Testing.

iii. Relationship with Developers
Big challenge. Requires very skilled tester to handle this relation positively and even by completing the work in testers way. For this tester also requires Good Communication, Troubleshooting and analyzing skill.

iv. Regression Testing
When a project goes on expanding the regression testing work simply becomes uncontrolled. The pressure to handle the current functionality changes, previous working functionality checks, and bug tracking.

v. Lack of Skilled Testers
Unskilled testers may add more chaos than simplifying the testing work. This results in incomplete, insufficient and ad-hoc testing throughout the Testing Life Cycle.

vi. Testing always under Time Constraint
When tester simply focuses on task completion and not on the test coverage and quality of work. There is a huge list of tasks that you need to complete within the specified time. This includes writing, executing, automating and reviewing the test cases.

vii. Understanding the Requirements
Sometimes testers are responsible for communicating with customers for understanding the requirements. Testers require good listening and understanding capabilities.

## *11. Discuss the test Automation for XP/Agile model.*

Agile Automation Testing in software development is an approach of using test automation in agile methodologies. The purpose of agile automation testing is to make the software development process more effective and efficient while maintaining the quality and time as well as resource consumption. Thus, the implementation of such a process requires a lot of coordination and collaboration between teams.

Agile methodology talks about doing away with laborious and tedious documentation so that new and innovative ideas could be implemented, and people could interact freely with each other so that more of these innovative and explorative ideas could be implemented.

The agile methodology by its own very definition is a sort of technique which is very helpful for responding to quick customer induced change requirements and which thus lends itself well to frequent changes during the overall development of the application.

## 12. *Explain features of J unit software testing tool*

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- Provides test runners for running tests.
- JUnit tests allow you to write codes faster, which increases quality.
- JUnit is elegantly simple. It is less complex and takes less time.
- JUnit tests can be run automatically, and they check their own results and provide immediate feedback. There is no need to manually comb through a report of test results.
- JUnit tests can be organized into test suites containing test cases and even other test suites.
- JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

## 13. *Explain different types of code coverage testing.*

Code coverage is a measure which describes the degree of which the source code of the program has been tested. It is one form of white box testing which finds the areas of the program not exercised by a set of test cases. It also creates some test cases to increase coverage and determining a quantitative measure of code coverage.

In most cases, code coverage system gathers information about the running program. It also combines that with source code information to generate a report about the test suite's code.

Following are major code coverage methods

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Toggle Coverage
- FSM Coverage

### ***Statement Coverage***

Statement Coverage is a white box testing technique in which all the executable statements in the source code are executed at least once. It is used for calculation of the number of statements in source code which have been executed. The main purpose of Statement Coverage is to cover all the possible paths, lines and statements in source code.

$$\textbf{Statement Coverage} = \frac{\textit{Number of executed statements}}{\textit{Total number of statements}} \times 100$$

***Decision Coverage***

Decision Coverage is a white box testing technique which reports the true or false outcomes of each boolean expression of the source code. The goal of decision coverage testing is to cover and validate all the accessible source code by checking and ensuring that each branch of every possible decision point is executed at least once. In this coverage, expressions can sometimes get complicated. Therefore, it is very hard to achieve 100% coverage.

$$Decision\ Coverage = \frac{Number\ of\ Decision\ Outcomes\ Excercised}{Total\ Number\ of\ Decision\ Outcomes}$$

***Branch Coverage***

Branch Coverage is a white box testing method in which every outcome from a code module (statement or loop) is tested. The purpose of branch coverage is to ensure that each decision condition from every branch is executed at least once. It helps to measure fractions of independent code segments and to find out sections having no branches.

$$Branch\ Coverage = \frac{Number\ of\ Executed\ Branches}{Total\ Number\ of\ Branches}$$

***Condition Coverage***

Condition Coverage or expression coverage is a testing method used to test and evaluate the variables or sub-expressions in the conditional statement. The goal of condition coverage is to check individual outcomes for each logical condition. Condition coverage offers better sensitivity to the control flow than decision coverage. In this coverage, expressions with logical operands are only considered.

$$Condition\ Coverage = \frac{Number\ of\ Executed\ Operands}{Total\ Number\ of\ Operands}$$

***Finite State Machine Coverage***

Finite state machine coverage is certainly the most complex type of code coverage method. This is because it works on the behaviour of the design. In this coverage method, you need to look for how many time-specific states are visited, transited. It also checks how many sequences are included in a finite state machine.