Test Automation Life Cycle

Test Automation Life Cycle (TALC) is a conceptual model used to implement, execute, and maintain an automation project (Fig. 2.1). It describes the stages involved in a test automation project from an initial feasibility study through maintenance of the complete automation project. It consists of the following phases: feasibility analysis phase, framework design and development phase, identification of business scenarios and business knowledge-gathering phase, test script design and development phase, test script execution and test result analysis phase, and maintenance phase.

FEASIBILITY ANALYSIS PHASE

Feasibility analysis phase is comprised of four stages: need analysis, Return On Investment (ROI) analysis, tool analysis, and proof of concept.

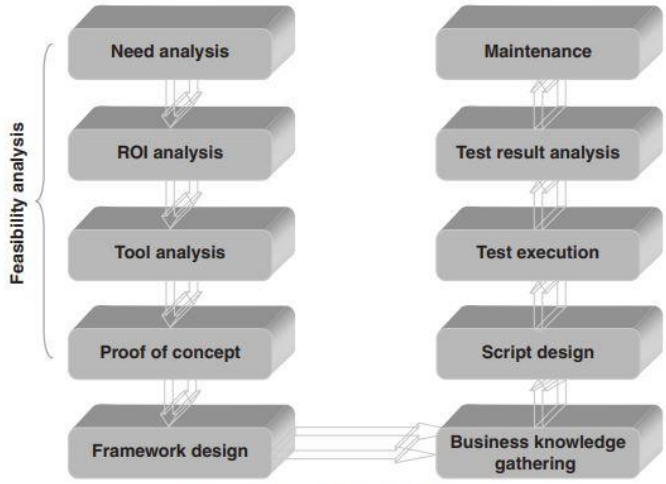


Figure 2.1 *Test automation life cycle*

Feasibility analysis phase is comprised of four stages: need analysis, Return On Investment (ROI) analysis, tool analysis, and proof of concept.

Need Analysis The first step before starting any automation project is to compare and analyze the expectations and the actual deliverables from the automation project. Project managers or automation consultants need to first list down the project requirements, management goals, expectations, etc.

ROI Analysis ROI analysis is used for evaluating the financial consequence of implementing an automation project. It compares the magnitude and timing of investment gains that are obtained from test automation directly with the magnitude and timing of investment costs.

Tool Analysis Tool analysis is done to select the tool for carrying out test automation. The automation tool that best suits the testing requirements and Application Under Test (AUT) needs to be selected. The automation tool should support AUT user interface.

Proof of Concept Proof of concept is done to practically test the automation tool against the project requirements. It is done to find out whether the selected tool supports the application GUI objects or not.

==========

2)Discuss briefly the three test automation development models ?

The three types of test automation development models that are widely being used as of today are

waterfall development model,

W-model,

agile development model.

Waterfall Development Model The waterfall model is a sequential design process where test automation starts once the application environment is stable. Generally, automation begins when the manual testing team has certified the build. The characteristics of waterfall development model are as follows:

● Automation starts after the application has become stable.

● A large part of application functionality is set to be automated in one go.

● Automation delivery is made once in several weeks/months after all the scripts have been developed.

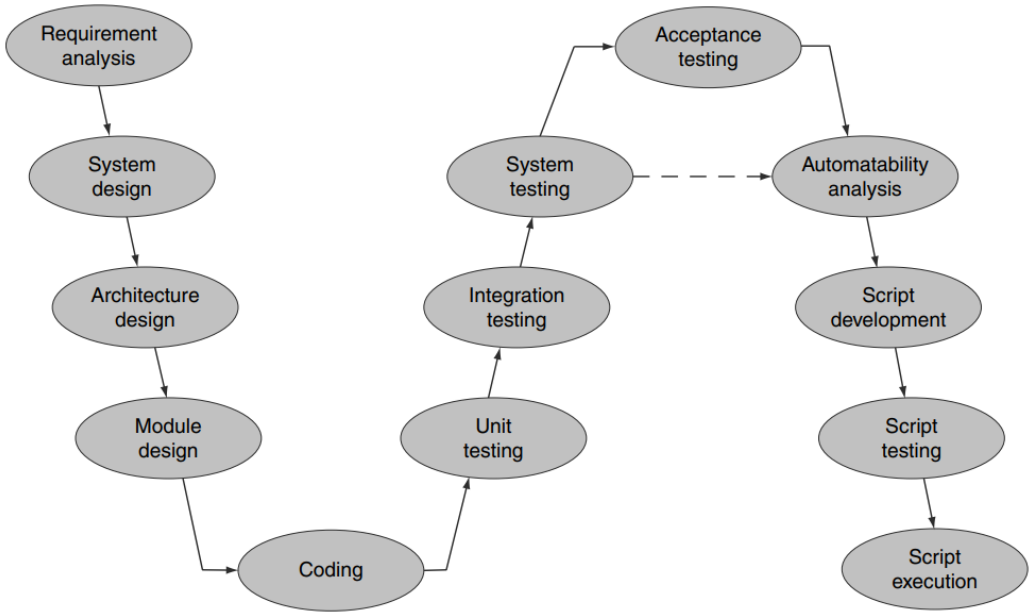


Figure 3.1 *Waterfall development model*

W-Model

W-model is an extension of waterfall model. In W-model, test automation process gets started as soon as the testing phase sets in. The automation processes are executed in parallel to the testing processes. The characteristics of W-development model are as follows:

- Automation planning is started when the test cases are being written.
- Automation development is started after the build is verified by the test team.
- A large part of application functionality is set to be automated in one go

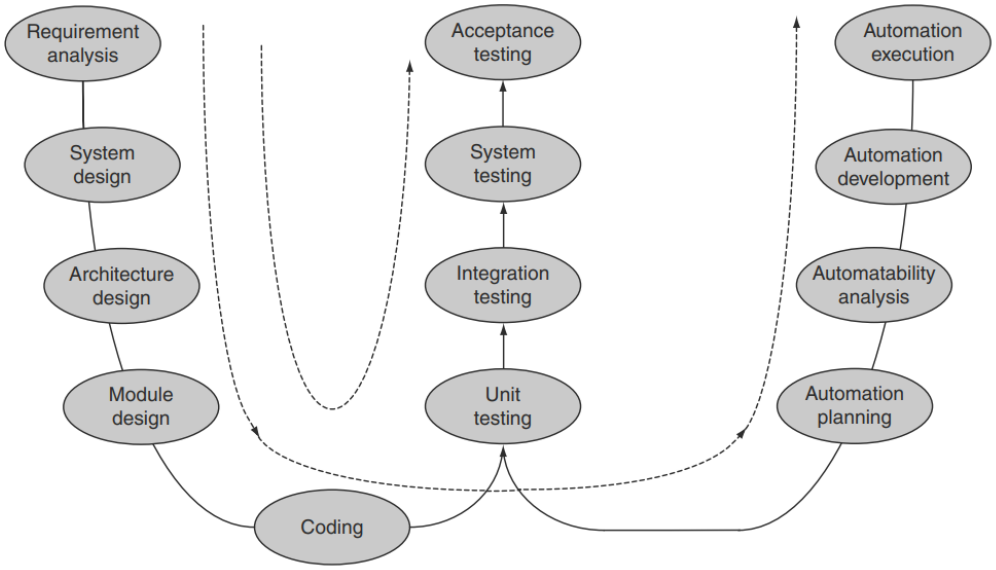


Figure 3.2 *W-development model*

Agile Development Model

Agile development model refers to a software development approach based on iterative development. In agile model, test automation tasks are split into smaller iterations or parts. Long-term automation plans are split into short-term achievable goals or milestones.

The test automation project scope and requirements are clearly laid down at the beginning of the development process. Plans regarding the number of iterations, the duration, and the scope of each iteration are clearly defined in advance. Each iteration is a short time 'frame,' which typically lasts for 2 to 4 weeks. The division of the entire project into smaller parts helps to reduce the project cost. The automated scripts are delivered at short time intervals. In case application development environment is also agile, test scripts are ready by the time build is moved to production. This implies test scripts can be used for regression runs right from the day one regression cycle. This saves a lot of manual effort, which was earlier required for executing manual regression runs.

==========

List various types of frameworks and explain any two framework types.

FRAMEWORK TYPES

There are various types of frameworks available as of today:

- Modular-driven framework,
- Data-driven framework,
- Keyword-driven framework,
- Hybrid framework,
- Business model driven framework, and
- Agile automation framework

Modular-driven Framework

Modular-driven framework requires creation of small, reusable, and independent scripts that represent a particular function of the application under test. These small scripts are then integrated in a hierarchical fashion to implement a test case. This framework applies the principle of abstraction to improve the maintainability and scalability of the automated test scripts.

Let us consider that we have www.irctc.com site as the application under test. Suppose we need to automate the following business scenarios:

1. Login → Book Ticket → Logout
2. Login → Cancel Ticket → Logout

Now, let us see how we can automate these scenarios using modular-driven framework. The first task is to split the business scenario into smaller reusable business functionalities.

Business functionalities—Login; Book Ticket—Plan Travel; Make Payment; Logout

Here, we observe that the specified business scenario has been split into four reusable business functionalities.

The next step is to design a driver that can sequentially call these reusable business functionalities to implement the business scenario.

Keyword-driven Framework

Keyword-driven framework is a framework where keywords are used to write test scripts step-wise in the form of table. This framework requires the development of data tables and keywords, which

are independent of the test automation tool used. Every keyword has its own controller code, which specifies which steps will be executed when the specified keyword procedure is called. Test case steps are written in step table for which a separate processor code is designed. In this framework, the entire test case is data driven including the test data. Since users cannot use logical loops in Excel sheets, this framework requires a new test script design (step table) for any logic change. In addition, in case of executing the same test script with different sets of data, step table needs to be duplicated. This results in huge repository of Excel sheets, whose maintenance again becomes very difficult.

Let us take an example of login to IRCTC site. For keyword-driven framework, the first task is to identify all the actions that can be performed by a user on the GUI end. Next step is to assign a 'keyword' for all these actions. Once all the 'keywords' have been identified a 'keyword map' table is to be created.

Keyword map defines all the keywords available for the test automation project. Table 4.1 shows an example of keyword map table.

Table 4.1 Keyword map

Keyword	Description
LaunchApp	Launch AUT
VerifyLogin	Check if login is successful
EditBox	Enter text into EditBox
Button	Click on button

==========

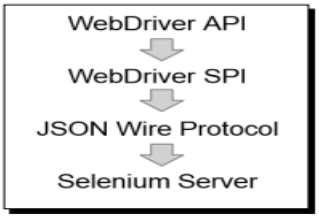
Discuss about the architecture of Selenium ?

Architecture

The WebDriver architecture does not follow the same approach as Selenium RC, which was written purely in JavaScript for all the browser automation. The JavaScript, in Selenium RC, would then emulate user actions. This JavaScript would automate the browser from within the browser. WebDriver on the other hand tries to control the browser from outside the browser. It uses accessibility API to drive the browser. The accessibility API is used by a number of applications for accessing and controlling applications when they are used by disabled users and is common to web browsers.

WebDriver uses the most appropriate way to access the accessibility API. If we look at Firefox, it uses JavaScript to access the API. If we look at Internet Explorer, it uses C++. This approach means we can control browsers in the best possible way but has the downside that new browsers entering the market will not be supported straight away like we can with Selenium RC.

Where that approach doesn't work we will then inject JavaScript into the page. Examples of this are found in the new HTML5.



The system is made up of four different sections.

WebDriver API

The WebDriver API is the part of the system that you interact with all the time. Things have changed from the 140 line long API that the Selenium RC API had. This is now more manageable and can actually fit on a normal screen. You will see this when you start using WebDriver in the next chapter. This is made up of the WebDriver and the WebElement objects.

```
driver.findElement(By.name("q"))
```

and

```
element.sendKeys("I love cheese");
```

These commands are then translated to the SPI, which is stateless. This can be seen in the next section.

WebDriver SPI

When code enters the **Stateless Programming Interface** or **SPI**, it is then called to a mechanism that breaks down what the element is, by using a unique ID, and then calling a command that is relevant. All of the API calls above then call down.

Using the example in the previous section would be like the following code, once it was in the SPI:

```
findElement(using="name", value="q")
sendKeys(element="webdriverID", value="I love cheese")
```

From there we call the JSON Wire protocol. We still use HTTP as the main transport mechanism. We communicate to the browsers and have a simple client server transport architecture the WebDriver developers created the JSON Wire Protocol.

`

JSON Wire protocol

The WebDriver developers created a transport mechanism called the JSON Wire Protocol. This protocol is able to transport all the necessary elements to the code that controls it. It uses a REST like API as the way to communicate.

Selenium server

The Selenium server, or browser, depending on what is processing, uses the JSON Wire commands to break down the JSON object and then does what it needs to. This part of the code is dependent on which browser it is running on.

==========

=

7. Write short notes on
i) CSS Selectors
ii) Page Objects

CSS SELECTORS :

We saw in the previous section that XPath selectors can offer your tests a lot of flexibility to find elements on the page.

CSS Selectors in Selenium are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selectors in Selenium is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.

CSS Selectors in Selenium have many formats, but we will only focus on the most common ones. The different types of CSS Locator in Selenium IDE

Tag and ID

Tag and class

Tag and attribute

Tag, class, and attribute

Inner text

When using this strategy, we always prefix the Target box with "css="

tag and id – CSS Selector

Syntax

```
css=tag#id
```

- tag = the HTML tag of the element being accessed
- # = the hash sign. This should always be present when using a Selenium CSS Selector with ID
- id = the ID of the element being accessed

tag and class – CSS Selector

CSS Selector in Selenium using an HTML tag and a class name is similar to using a tag and ID, but in this case, a dot (.) is used instead of a hash sign.

Syntax

```
css=tag.class
```

- tag = the HTML tag of the element being accessed
- . = the dot sign. This should always be present when using a CSS Selector with class
- class = the class of the element being accessed

tag and attribute – CSS Selector

This strategy uses the HTML tag and a specific attribute of the element to be accessed.

Syntax

```
css=tag[attribute=value]
```

- tag = the HTML tag of the element being accessed
- [and] = square brackets within which a specific attribute and its corresponding value will be placed
- attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID.
- value = the corresponding value of the chosen attribute.

lı) Page Objects

Page objects in Selenium are a design pattern that creates an object repository for web UI elements1. The advantage of this model is that it reduces code duplication and improves test maintenance12. Under this model, for each web page in the application, there should be a corresponding page class that serves as an interface to that page1. The page class will identify the web elements of that page and also contains page methods which perform operations on those web elements

A short note on page objects in Selenium could be:

Page objects in Selenium are a way of organizing the test code and page-specific code, such as locators and layout

Page objects in Selenium help to create a single repository for the services or operations the page offers rather than having these services scattered throughout the tests

Page objects in Selenium make the code more readable, maintainable, and reusable

Page objects in Selenium separate the test methods and verification methods from the page elements and operations

Page objects in Selenium can be integrated with different tools and frameworks for functional or acceptance testing

==========

Explain how to add filters to the name ?

Adding filters to the name in Selenium means using additional attributes or conditions to locate an element that has a name attribute. This can be useful when there are multiple elements with the same name value on a web page, and we need to find a specific one1.

There are different ways to add filters to the name in Selenium, depending on the locator strategy used. Some of the common ways are:

Using CSS selectors: We can use a dot (.) for class, a hash (#) for id, or square brackets ([]) for other attributes along with the name attribute to create a CSS selector2. For example, css=input [name=lastName] will find an input element with name attribute value as lastName2.

Using XPath: We can use various XPath axes, functions, and operators to create complex expressions that can filter elements by name and other criteria3. For example, //input [@name='verifybutton' and @value='chocolate'] will find an input element with name attribute value as verifybutton and value attribute value as chocolate3.

Using filters in Selenium IDE: We can use filters in Selenium IDE to add extra information to the name locator. A filter is a semicolon-separated list of attribute-value pairs that can be appended to the name locator1. For example, name=verifybutton value=chocolate will find a button element with name attribute value as verifybutton and value attribute value as chocolate1.

1 MARKS :

What is the significance of ROI Analysis?

ROI analysis is a part of the life cycle of test automation because it helps to evaluate the financial benefits and costs of implementing automated testing12. ROI analysis is a way of measuring the return on investment (ROI) of test automation, which is the ratio of savings to investment12.

The significance of ROI analysis is that it helps to:

- Justify the initial and ongoing expenses of test automation, such as tools, frameworks, scripts, and maintenance.
- Compare the efficiency and effectiveness of test automation with manual testing, such as time, quality, and coverage.

Specify different types of Test Automation Frameworks

Linear scripting framework

Modular testing framework

Data-driven testing framework

Keyword-driven testing framework

Hybrid testing framework

What is Selenium IDE?

Selenium IDE is a record and playback tool for creating and executing automated tests for web applications. It is a browser extension that provides a simple interface for creating and running test cases without the need for programming knowledge

OR

Selenium IDE is a browser extension that allows users to record and playback their interactions with web applications using Selenium commands12. It is an easy-to-use tool that does not require any programming skills and can be used for learning Selenium syntax or creating simple test cases

What the difference is between verify and assert?

Verify and assert are two types of validation methods used in software testing, especially in test automation frameworks12. The difference between them is that:

Assert: The test will stop if the assert fails. This means that the test execution will be aborted and the test result will be marked as failed. Assert is used when the validation is critical and further execution is not possible or meaningful without passing the assert12.

Verify: The test will continue even if the verify fails. This means that the test execution will not be interrupted and the test result will be marked as failed only at the end of the test. Verify is used when the validation is not critical and further execution can still be performed with or without passing the verify12.

State any two principles of agile automation.

Two principles of agile automation are:

Customer satisfaction by delivering the software early: This principle is based on the agile manifesto, which states that the highest priority is to satisfy the customer through early and continuous delivery of valuable software. Agile automation supports this principle by enabling frequent and reliable delivery of working software through automation tools and frameworks
.
Continuous attention to technical excellence and good design: This principle is also based on the agile manifesto, which states that continuous attention to technical excellence and good design enhances agility. Agile automation supports this principle by ensuring that the automation code and scripts are well-designed, maintainable, reusable, and scalable.

What are screen components?

Screen components in software testing are the graphical user interface (GUI) elements that are used to create and display a software application or system on a screen. They include various types of controls, menus, icons, buttons, text fields, progress bars, etc. that allow users to interact with the software or system12. Screen components can be tested using different tools and frameworks, and can be customized according to the needs and preferences of the users12.
Some examples of screen components in software testing are:

Title bar: This is the horizontal bar at the top of a window that displays the name of the software or system3.
Horizontal button bar: This is a row of buttons that provide quick access to common functions or settings3.

Close button: This is a button that allows users to close or exit a window or application3.
Gateway of Tally: This is a menu that displays the options and features of a software application called Tally ERP 93.

Specify the fields of screen component table ?

The attributes may include data type, length, format, validation rules, default value, and other properties

State the objectives of automation testing.?

The objectives of automation testing are to reduce testing cost and time, reduce redundancy, speed up the testing process, improve quality, improve test coverage, and reduce manual intervention[1]. Automation testing can also help enhance software quality after each iteration[2]

What are the key factors of Test Automation.?

The key factors of test automation include a goal-oriented strategy, defining the test approach, setting up tools and test environment, designing test scripts and writing test cases, and identifying modules or components of the software under test that can be easily tested in an automated manner[1]

Specify any four guidelines to be followed for test automation.

Identify the right test cases for automation: Not all test cases are suitable for automation. It is important to identify the right test cases that can be automated.

Choose the right tool: There are many tools available for test automation, and it is important to choose the right tool that meets your requirements.

Plan and design your tests carefully: Test planning and design are critical to the success of test automation. It is important to plan and design your tests carefully to ensure that they are effective and efficient.

Maintain your tests: Test maintenance is an important aspect of test automation. It is important to maintain your tests regularly to ensure that they remain effective and efficient.