

Automation Strategy and Guidelines

February 2022

**Contents**

[Test Automation 3](#_Toc96518037)

[What is Test Automation? 3](#_Toc96518038)

[Test Automation Strategy 3](#_Toc96518039)

[Test Automation Pyramid 3](#_Toc96518040)

[Goals of Test Automation 4](#_Toc96518041)

[Test Automation Approach 5](#_Toc96518042)

[Test Automation Scripting and Execution 6](#_Toc96518043)

[Scripting 6](#_Toc96518044)

[Peer Review 6](#_Toc96518045)

[Execution 7](#_Toc96518046)

[Test Automation Workflow 8](#_Toc96518047)

[Automation Testing Team Responsibilities 8](#_Toc96518048)

[Test Coverage 8](#_Toc96518049)

[What is Test Coverage? 8](#_Toc96518050)

[Test Coverage Objectives 9](#_Toc96518051)

[Test Automation Coverage Approach 9](#_Toc96518052)

[Test Automation Tools 10](#_Toc96518053)

[What are Test Automation Tools? 10](#_Toc96518054)

[Types of Test Automation Tools 10](#_Toc96518055)

[Test Automation Tool Selection Criteria 11](#_Toc96518056)

[Selection Criteria 11](#_Toc96518057)

[Selection Checklist 11](#_Toc96518058)

[Popular Test Automation Tools and Frameworks 12](#_Toc96518059)

[Selenium 12](#_Toc96518060)

[BDD Frameworks 13](#_Toc96518061)

[API Testing Tools 13](#_Toc96518062)

Test Automation

What is Test Automation?

Test automation is the process of test execution with little or no human interaction. This method helps the Testing Team to achieve greater speeds, reliability and efficiency. The manual tests are replicated with an automation script which reduces the efforts of the testers. There are a variety of test automation approaches and strategies which can be employed and there are lots of automation tools in the market which help us to achieve these strategies.

Test Automation Strategy

Deciding on the right automation strategy is vital for assuring proper testing, minimizing development cost, and delivering the lowest possible ongoing maintenance expenses. This includes identifying tests to automate and the decision of when the test cases should be developed during the sprints. The strategy also defines the roles for designing and reviewing test cases.

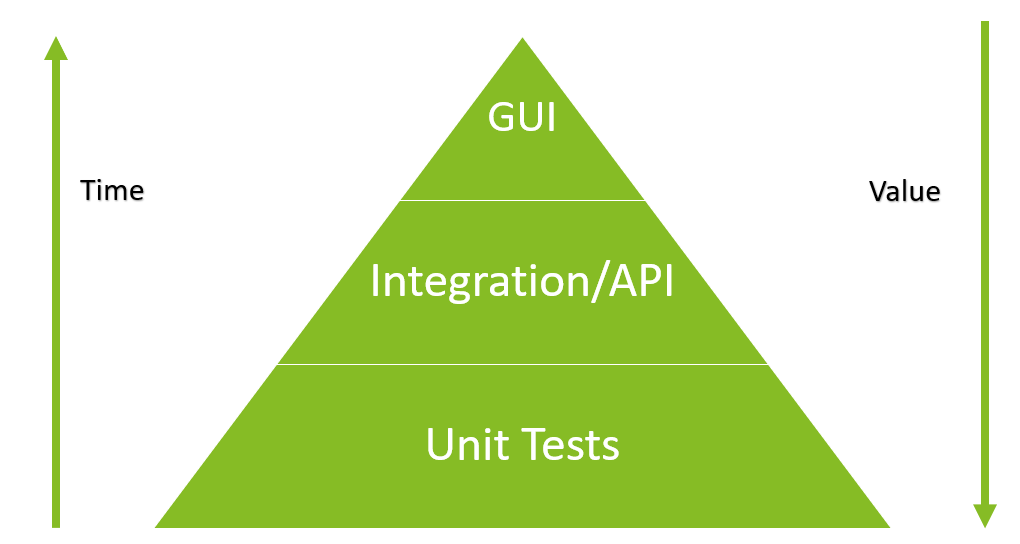
When choosing a test automation strategy, there are three areas to consider: *Processes, Technology, and Roles*

1. **Process –** Automation implementation must always be a well-defined and structural process.
2. **Technology –** The technologies used in the application is analyzed and the automation tools that serves as a best fit is selected.
3. **Roles –** Roles and its corresponding responsibility in the test automation team is defined in this phase. All the members must be clear about each one’s role and detail process.

Test Automation Pyramid

The testing pyramid is a concept that groups software tests into three different categories. This helps to ensure higher quality, reduce the time it takes to find the root cause of bugs, and build a more reliable test suite.

Essentially, the test automation pyramid lays out the types of tests that should be included in an automated test suite. It also outlines the sequence and frequency of these tests. The aim is to offer immediate feedback to ensure that code changes do not disrupt existing features.



1. **Unit Tests –** They test individual components or functionalities to validate that it works as expected in isolated conditions. There are several scenarios in unit tests – happy path, error handling, etc. These are run every time a new feature is added. Consequently, developers receive immediate feedback on whether individual features are working as they are meant to.
2. **Integration Tests –** Integration tests are the second layer of the test automation pyramid. Fundamentally, they test how a feature communicates with external dependencies such as a call to a database or web service, the software needs to communicate effectively and retrieve the right information to function as expected.
3. **API Tests –** API automation lets you validate the core logic without being dependent upon the UI. With API automation, testing of an application can start early, even without a UI. This helps to identify and fix issues early in the development lifecycle which would otherwise be expensive to fix when identified during UI testing.
4. **End-to-End Tests/ UI Tests –** At the top of the pyramid are the end-to-end tests or UI tests. These ensure that the entire application is functioning as required. These tests verify that the application is working flawlessly from start to finish. These tests are at the top of the testing pyramid because they usually take the longest to run. They can also be fragile since they must test a large variety of user scenarios.

Goals of Test Automation

At this stage, scope of testing is defined. The deciding factor for any test to be automated is the one that runs often, is a mundane task, time-consuming, and requires a huge amount of data to perform the stipulated task.

Key pointers for selecting and prioritizing test cases for automation:

* Critical from the business point of view
* Carry a high risk of failure
* Must be executed repeatedly, for example, log-in
* Involve large data sets or require operating through multiple data sets
* Tasks that require testing over multiple browsers/ devices/ OS/ environment/ hardware/ configurations
* Result in a clear Pass/Fail
* Functionalities that are stable

Criteria to exclude test cases from test automation:

* Tests that will be executed a single time (except for the data-driven and regression tests)
* UX tests
* Tests without clear Pass/Fail results
* Tests based on visual perception
* Testing raw and unstable functionality
* Ad hoc and exploratory testing where you check the system based on your knowledge of its internal workings and experience
* Testing containing anti-automation features (e.g., CAPTCHA)

The challenges in test automation resides in surviving the fast and frequent UI layout and functionality changes. Even the smallest modification may require a rewrite of substantial number of tests. One of the ways to minimize this inconvenience is to **ensure maximum test code reuse**. Success in test automation requires meticulous planning and creating standardized high-level application-specific libraries.

Test Automation Approach

Testing approach represents a set of rules and guidelines to follow scripting and executing tests like coding standards, practices, processes, reporting etc. Planning the test approach includes retrospection of the current process, finalizing and deciding the new test approach as well as determining the level of testing, roles, and responsibilities of the team members. The team works together with the goal of automation and decides on a testing method that provides the maximum value when automated. Different testing methods can be employed for different testing levels.

**Five main test approaches for automation:**

1. **Linear**: The most basic framework. Testers perform each step manually and record them with an automation tool which can be executed multiple times
   1. Pros: Simple, fast, and requires no planning.
   2. Cons: Difficult to reuse and maintain
2. **Module-Based**: Testers divide a system into several independent modules and write scripts to test each one separately.
   1. Pros: Modularity, low maintenance, scalability.
   2. Cons: Tests needs to be re-worked to execute different data set.
3. **Test Library Architecture**:  After every test case is recorded with an automation tool, common steps are grouped into functions to be collated as a library, which can be reused multiple times.
   1. Pros: Modularity, low maintenance, high reusability, and low cost.
   2. Cons: Time-consuming, test data is embedded into the scripts, requires better skillset.
4. **Data-Driven Testing**: In this approach, data is separated from automated test scripts and stored in external files. Tests are executed with different data sets without re-scripting them.
   1. Pros: Data and test scripts are independent, easy to reuse, flexible and low maintenance.
   2. Cons: Time consuming, and Require good coding skillset
5. **Hybrid Approach**: Hybrid approach is a combination of better parts of every approach mitigating the weaknesses of different frameworks. It is an approach to produce accurate test results as it favors flexibility of different applications.
   1. Pros: Flexible, reusable, and adaptable, can be integrated with CI pipelines, Granular reporting
   2. Cons: Test automation expertise is needed

Considerations when selecting a test automation framework and approach: -

* Technologies used in the application
* Associated costs
* Available talent
* Requirements

Prioritizing on a smoke test that will quickly execute for each new build is a low-risk and good approach to get started with test automation. This strategy allows teams to adapt the framework gradually as the need of the application is realized.

Test Automation Scripting and Execution

Scripting

Test Automation utilizes scripts which go through a development phase like other pieces of code. Some of the tools used for test automation require programming skills for the development and maintenance of the scripts. A test automation script consists of the following components,

* **Test Data:** It is the data which is used as an input in the test execution and is defined before running the tests. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.

Test Data is generated as follows,

* + Manually
  + Mass copy of data from production to testing environment
  + Mass copy of test data from legacy client systems
  + Automated Test Data Generation Tools
* **Test Method:** This is the set of instructions to test a particular scenario in the application. The elements are identified/located in the application and the set of instructions scripted are performed on the elements. The output is then stored and verified based on the assertion provided.
* **Assertions:** These are the conditions used to validate the behavior of the application when some sets of instructions are performed on the application. These are simply validations that determine if the test method succeeded, and the behavior of the application is as expected. These are used to pass or fail the tests.

Peer Review

Peer review, a review technique is a static white-box testing which are conducted to spot the defects early in the life cycle that cannot be detected by black box testing techniques. It is a way of evaluating work performed by a co-worker.

**Benefits of Peer review:**

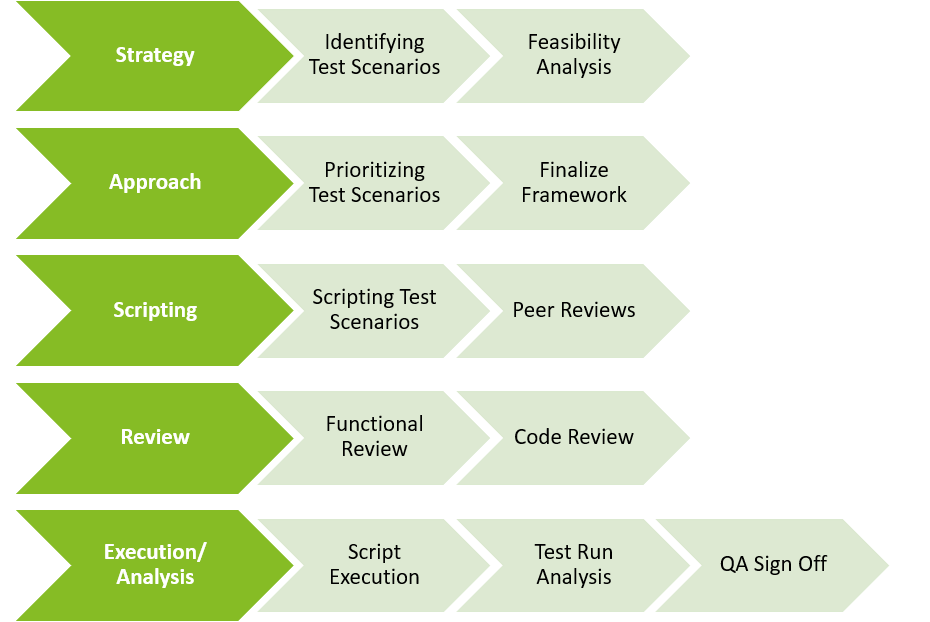
* **Increase in the quality of the work product -** Quality is addressed through Peer review. As a result, the process progresses with an inquisitive approach thus enabling the possibility of least number of defects.
* **Steady work progress -** A milestone is defined prior commencing the process of peer review. Hence when the process begins, the team members gear up for achieving the final objective.
* **Enhanced collaboration between teams -** The teams working in association with each other helps them exchange ideas, take opinion of their fellow team members on similar kind of problems. This ensures enhancement of knowledge which can help alleviate challenges, effortlessly.
* **Increased Learning -** When team members exchange ideas and opinions within the team or with another team, it gradually helps in developing a great deal of knowledge about a subject.
* **Cost effective -** Imbibing peer review as a part of software development life cycle helps to curb cost in terms of outsourcing the application under development for review to some external resource, purchasing or renewing tool licenses.

Execution

Test Execution phase is an important phase of the testing lifecycle. This phase is the part of the lifecycle in which the actual testing takes place for the application. Here is the list of things in the test execution phase,

* **Build** – The scripts which are pieces of code which have been designed and implemented is packaged into a deployable solution. The solution is then deployed on a Test environment and is the first step for Test Execution.
* **Environment** – All environments in the development, testing and production are as identical as possible in nature and stable to replicate real time conditions. Planning of test cycles, test cases and predicting expected testing time is necessary for deciding on the test environment.
* **Smoke and Sanity** – Test Execution starts with these tests which are considered baselines for an application. Sanity tests is a checkpoint to assess if testing can proceed for the build. If the sanity test fails, the build is refused to save the costs and time complexity in more severe testing.
* **Test Cycle** – Test cycle begins here with a round of tests to cover all the functionalities and in some cases, this is Regression Testing. The results of this execution will determine the next steps of testing and exploratory testing.
* **Exploratory Testing** – This is a simultaneous learning, test design, and test execution process. This testing process contains test planning, analysis, design and test execution which are done instantly. This type of testing is done further based on the results of the other tests to isolate any functionality and fully test it.
* **Test Result Analysis** – The last phase of this to collate the test results and analyze the results to bring out the defects of the application.

Test Automation Workflow



Automation Testing Team Responsibilities

|  |  |
| --- | --- |
| **Owner** | **Responsibilities** |
| Business Analyst  Test Lead/ Test Manager | * Identification of Test Cases for Automation |
| Test Engineer | * Test Case Scripting * Creation of Test Data * Environment setup * Test Execution and Maintenance |
| Test Lead | * Code Reviews and Enhancements |

Test Coverage

What is Test Coverage?

Test Coverage is the technique to assess the quantity of testing by the test plan. It is a metric widely used in software testing to measure and improve the quality of test sets. It has three type of metrics code-level metrics, feature-testing metrics and application-level metrics.

* **Code Level Metrics:** Code metrics is a set of software measures that provide developers better insight into the code they are developing. Code level metrics cover the test execution coverage percentage.
* **Feature Level Metrics:** Requirement’s coverage is used to determine coverage of test cases as per the software requirements. Feature testing metrics cover the requirements coverage. These metrics are used to map the features being tested and the number of tests that aligned with a requirement.
* **Application-Level Metrics:** Application-level metrics cover the defect densities. These help in identifying functionalities which require retesting.

Test coverage can be evaluated through different types of testing such unit, functional, acceptance testing. It can also be represented using the Test Coverage Matrix (TCM) which ensures that functionality is verified in all possible combinations. Fundamentally teams with higher test coverage not only see fewer bugs in production, but also better collaboration between Testing Team and developers.

Test Coverage Objectives

Test Coverage tends to correlate closely with the quality of the product. The objectives of test coverage help in achieving a higher test coverage which means fewer bugs are present in the production. The objectives are defined as follows,

* **Defect Elimination:** The objective here is to eliminate the gaps and defects at early stages and to accomplish a smoother testing cycle, keeping the project scope intact.
* **Effective Test Automation Plan:** The objective here is to achieve maximum coverage with limited test cases which makes up the effective test plan strategy.
* **Maximizing Return on Investment:** Test Coverage will have a major impact on the testing phase which translates to a higher ROI. The objective here is to create an impact on ROI by implementing test coverage that makes processes achieve a higher quality and effectivity.
* **Test Automation Approach:** Having a higher test coverage does not translate to higher quality due to the quality of the test automation. The objective here is to plan the test automation approach with the right metrics and apply the test coverage on it rather than following the approach designed around test coverage.

Test Automation Coverage Approach

Test Automation Coverage should enhance the efficiency of a continuous delivery practice. The following steps will define an approach to improve test automation coverage –

**Step 1 – Application of Test Coverage metrics:** Test coverage metrics is applied to automated test cases to define a coverage in terms of test automation. This helps in getting an understanding of the efficiency of the automation test plans and automation scripts.

**Step 2 – Mapping and Assigning Priority:** The automated test cases can be mapped to requirements of the application. The user’s scenarios can also be mapped to automated test cases. These are then ranked by priority which helps in distinguishing critical functionalities. These steps help in analyzing gaps in the automated test cases.

**Step 3 – Automating the rest:** The gaps analyzed from the previous steps can be put in the backlog and implemented accordingly to ensure that the automated tests have an improved test coverage. Here are some of the factors which contribute to the gaps in test automation,

* Coverage of Important Functionalities – Test automation must cover all critical functionalities.
* Coverage of the environments such as browsers and operating systems – Tests must be executed on different browsers and operating systems.
* User Experience in real life conditions – Tests must mimic users’ scenarios and conditions in real life.
* Consistency – Test automation must be consistent in running all the tests in the automation every time without filtration.

**Step 4 – Automation Tools:** Tools are an important factor in test automation. The implementation of test automation using feature-rich tools can improve test coverage and make the test automation more reliable. The scenarios which are to be retested are the ones to be automated with tools to reduce the time of regression tests and for faster releases.

**Step 5 – Maintain and Expand:** The application can become more complex in its features and larger in scale. The test automation implemented should be expanded with new test cases and scenarios. In an agile approach to software development, or in a continuous integration / continuous delivery (CI/CD) pipeline, teams often must balance their goals for improving test coverage with their goals for releasing new features fast enough to keep up with the market needs.

Test Automation Tools

What are Test Automation Tools?

These are tools which help in achieving test automation. These tools can help to improve test coverage and make the test automation effective. There are a wide variety of tools which can be used for automation. They might differ in the types of application they test which can be web, desktop and mobile, in the way the test cases are set up by writing code using a scripting language, writing code in a full programming language, recording steps performed using a GUI and in their licenses, which are free, freemium, commercial and many other factors.

Types of Test Automation Tools

The tools available can be categorized as follows,

* **Open-Source Automation Tools:** These tools are free platforms that allow users to access and use their source code. The source code can be customized to suit the testing needs. Most of the teams utilize open-source tools to reduce project costs and customize the test automation according to the project requirements.
* **Commercial Automation Tools:** These are built especially for commercial purposes with subscription plan models. Users purchase a license to use these tools. These tools have technical support and premium features to help with the testing experience. These factors help in completing the whole testing process for companies.
* **Frameworks:** These are projects which generally combine features of multiple tools and are customized pieces of software to accomplish complex testing processes and testing in a variety of environments. These are custom built for a specific project and are more complicated to setup and use relatively. They require comprehensive coding skills and experience to implement test automation.

Test Automation Tool Selection Criteria

Selection Criteria

* **Breadth – Expected Usage:** A deep understanding of project requirements is the prerequisite to tool selection. Project requirements like project type (web, mobile, desktop), scope of the project, existing strength on a language are some of the factors that determine the path to be followed while covering the tools available in the market.
* **Impact – Degree of Efficiency:** The tool selected should be effective in reducing the repetitive tasks and increase the test coverage. The development & maintenance of test scripts should be as simple as possible to decrease the human and time resource utilization. The potential for automation should cover all the requirements of the project and the scope of the project to create an overall impact on the project.
* **Business Value – ROI:** The tool’s efficiency will determine the business value it adds to the project. The automation tool should improve productivity, project margins and reduce the costs. The tool selected should cover these factors to ensure that a higher and strong ROI is built.
* **Feasibility – Reducing Manual Efforts:** A feasibility analysis of the potential scenarios for automation in a project should be an integral part of the testing strategy. A tool which can automate most of the scenarios and reduce the amount of work hours for testing is required.
* **Adoptability – Ease of Automation:** The development time for scripts must be made shorter in order to have more times for other processes like planning and strategy. The tool selected should be simple to adopt and the rate of conversion of test scenarios to automation must be high. The ease of development using the tool is an important factor to consider during selection.
* **Sustainability – Ease of Maintenance:** The test automation strategy should consider the maintenance of test scenarios and test plans as the strategies are planned for the long run. As test automation increases in the project, the test cases will require maintenance and constant monitoring. The tool selection should include this factor to reduce the effort in maintenance which will shift the focus of the team to automate more test scenarios rather than fixing the existing ones.

Selection Checklist

A checklist based on the criteria for selection of a tool is as follows,

* + **Test Development** – what type of tests can be developed
    - Cross-Platform
  + **Application Under Test** – what type of application is being tested
* Web Application
* Mobile App
* Desktop
  + **Scripting Language Support** – which programming languages the tool supports
  + **Continuous Integration Support** – which platforms does the tool support
    - Jira
    - Azure DevOps
  + **Extended Features**
    - Image Based Testing
    - Object Storage and Maintenance
    - Test Reporting
  + **Cost** – the costs to setup the tool
    - Open Source
    - Subscription Based - Recurring
    - One Time Subscription - Fixed
  + **Support**
    - Documentation
    - Community Forums
    - Product Support/ Technical Support
  + **Usability**
    - Ease of Test Script Development
    - Test Scheduling
    - Framework Support – Keyword, Data Driven, Hybrid

Popular Test Automation Tools and Frameworks

Selenium

There are several tools in the market but the most popular one is Selenium. It is a suite of tools consisting of components, each one of them playing an explicit role in the development. **Selenium WebDriver** is the web-driver API which contains a wide array of methods to implement test automation on web applications.

**Test Development –** Selenium has a platform independent approach and works on a variety of environments such as different browsers or operating systems.

**Application Under Test** – Selenium can be used to test web applications.

**Scripting Language Support** – Selenium supports scripting in various languages such as Java, C#, Python, etc.

**Continuous Integration Support** – Selenium supports continuous integration on Jenkins and on other platforms with customizations. Azure dev ops also support selenium-based tests but requires other frameworks such as MS Test for Integration.

**Costs** – Selenium is open source and is customizable and hence it gains popularity in the market.

**Support –** Selenium does not have a support forum but because of its wide usage across the community there are a lot of third-party forums and communities which have a lot of documentation and support for Selenium.

**Usability –** Selenium requires programming skills but is easy in terms of development. It is relatively easy to maintain and is developer friendly.

BDD Frameworks

BDD test automation frameworks read executable specifications written in plain text and validate that the software meets its requirements. A BDD testing framework like Cucumber and SpecFlow have enabled non-technical product owners to define application behavior as human-readable text.

**SpecFlow:** SpecFlow is the most popular BDD framework for Microsoft .NET languages. It is a free and open-source framework, and it requires a separate unit test runner (e.g., Unit or MS Test) in order to run scenarios.

**Cucumber:** Cucumber is a popular BDD framework which verifies whether the software conforms with the specification and generates a report indicating success or failure for each scenario. It works on syntax rules called Gherkin. Gherkin is a set of grammar rules that makes plain text structured enough for Cucumber to understand.

API Testing Tools

API testing effectively validates the logic of the build architecture within a short amount of time. The API testing is performed at the business layer, in which business logic processing is carried out and all transactions between the user interface and database layers occurs at this layer. There are two variants of tools for API Testing which are as follows,

**Frameworks:** These are libraries which help to perform API Testing. Frameworks are customized libraries which are made utilizing the API Testing Libraries such as Rest Assured and Rest Sharp.

* REST Assured – These are JAVA libraries used for API Testing and these integrate well with Maven. Rest Assured has methods to fetch data from almost every part of the request and response even if they contain a complex JSON structure.
* REST Sharp – These are like Rest Assured libraries but are built for C# programming language.

**Standalone:** These are tools for API Testing which can be used independently without relying on several libraries. Postman is such a tool which has an interface to allow users to extract web API data. It comprises of built-in tools, collections, and workspaces. It supports various formats, including RAML and Swagger.