International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

Automation Landscape: A Logical Analysis from Framework Absence to Optimal Selection

Rohit Khankhoje

Independent Researcher
Email: *rohit.khankhoje[at]gmail.com*

Abstract: ***In the ever-changing landscape of software automation, the decision to choose the most appropriate framework holds great significance as it has the potential to greatly impact the effectiveness of testing and quality assurance processes. This paper aims to address the initial challenges that organizations may encounter when they lack a structured automation framework. It emphasizes the importance of recognizing this absence and the opportunity it presents for improvement. Furthermore, we will examine the key factors involved in transitioning from an absence of framework to selecting an optimal automation framework. Throughout this exploration, we will underscore the importance of evaluating organizational requirements, considering compatibility with technology, and assessing scalability. We will delve into the various types of frameworks that are available and discuss how to align them with specific aspects of projects. Additionally, we will discuss the necessity of adaptability in a rapidly evolving technological landscape. In the pursuit of efficiency, reliability, and cost-effectiveness in automation efforts, the process of selecting a strategic framework becomes increasingly crucial. This paper provides insights into the logical analysis required to bridge the gap from the absence of a framework to making an optimal selection. By doing so, it equips professionals with the knowledge needed to make informed decisions and maximize the potential of their automation initiatives. For those seeking clarity in the complex realm of automation framework selection and implementation, this paper serves as a valuable resource.***

Keywords: Test automation, Automation, Traditional Framework, Keyword driven framework, Software testing

1. Introduction

In the contemporary and rapidly evolving technological environment, the integration of automation has emerged as an indispensable element within the realm of software testing and quality assurance processes. As diverse organizations across various industries strive to optimize efficiency, minimize human error, and expedite time-to-market, the significance of automation within their operations has become paramount. However, while the adoption of automation is a strategic imperative, the selection of the most appropriate automation framework is not a trivial undertaking. It is a decision that possesses the potential to greatly influence the success of testing endeavors.

The paper, entitled "Automation Landscape: A Logical Analysis from Framework Absence to Optimal Selection," constitutes a comprehensive paper that is meticulously crafted to navigate the intricate domain of automation framework selection. This paper duly acknowledges that numerous organizations, during the nascent stages of automation implementation, find themselves bereft of a structured framework. In order to address this challenge, our paper unveils a logical and strategic analysis that guides organizations from a state of framework absence to a position of optimal selection (Bajpai, 2011). The absence of an automation framework should not be considered a limitation, but rather an opportunity for growth and enhancement. Some organizations have no framework but have a set of scripts which is called non structured automation.

Our exploration encompasses the critical factors that exert influence over this decision, encompassing organizational requirements, technological compatibility, scalability, and industry-specific needs. it underscores the significance of adaptability within a dynamic technological landscape where change is the sole constant.(Bhondokar et al., 2013)

This paper empowers professionals and organizations to make well-informed decisions, ultimately ensuring thattheir automation endeavors align seamlessly with their objectives and yield tangible outcomes. It furnishes a holistic and practical perspective on automation framework selection, positioning itself as a valuable resource for navigating the intricate journey from framework absence to optimal selection. In the subsequent sections, we will delve into the key considerations and logical analyses that underlie this transformative process.

2. Background

The potential benefits of automation are accompanied by inherent difficulties. One of the pivotal determinations that entities in these fields must confront is the adoption of the appropriate automation framework. This decision plays a crucial role in determining the efficiency, expandability, and enduring sustainability of automation interventions.

Organizations may face a multitude of problems if they do not have a meticulously chosen structure in place. Difficulties emerge when automation initiatives lack coherence, uniformity, and flexibility. These deficienciescan result in escalated operational expenses, diminished quality assurance, and ultimately impede an organization's capacity to remain competitive in a rapidly changing technological environment. This paper endeavors to examine the crucial choice of framework selection, demonstrating the progression from the lack of a framework to the rational examination that forms the basis for the selection of the most effective solution.

By illuminating the complications encountered in the

absence of an automation framework and explaining the factors that impact the selection of such a framework, this research paper offers a comprehensive perspective on the panorama of automation. It underscores the importance of adopting a methodical approach in fully exploiting the potential of automation.

3. Journey to Automation

1) *Initial State: Feasibility of Automation-* The commencement of each endeavour towards mechanisation originates from an examination of its viability. Preceding delving into the technical facets, it is crucial to evaluate whether mechanisation is the appropriate selection for the assigned undertaking. (De, 2009) This preliminary phase encompasses analysing the existing procedures, identifying recurring and laborious activities, and approximating potential advantages, such as temporal and monetary efficiencies. After the establishment of the viability of mechanisation, the journey advances into thecustomary methodology.
2) *Traditional Approach: Manual Efforts and Scripting-* The subsequent stage in the conventional methodology involves the advancement of scripts. These scripts are frequently generated utilizing programming languages such as Python, Java, or other scripting languages. Initially, scripting concentrates on the automation of small, manageable segments of the workflow. These scripts are typically customized to specific tasks and serve as fundamental components for automation.
3) As more procedures are automated, the scripting endeavors expand to encompass error handling, data validation, and reporting. Nevertheless, difficulties emerge as the quantity of scripts increases.
4) Maintenance becomes a significant undertaking, and ensuring the scalability, robustness, and reusability of scripts becomes a top priority.
5) *Framework-Based Approach: Scaling and Efficiency-* In order to tackle these obstacles, the expedition transitions towards an approach centered on a framework. During this stage, a framework assumes the role of a fundamental element for the process of automation.

Transitioning to a framework-based methodology entails constructing or embracing a framework that is in harmony with the objectives of the organization. Frameworks can be uniquely built using programming languages, libraries, and open-source tools. Alternatively, establishments can exploit commercial testing frameworks that are tailored to specific industries.

As the expedition advances towards the framework-based methodology, endeavors in automation become more effective and sustainable. Teams are equipped to confront intricate tasks, tackle scalability issues, and fully realize the potential of automation.

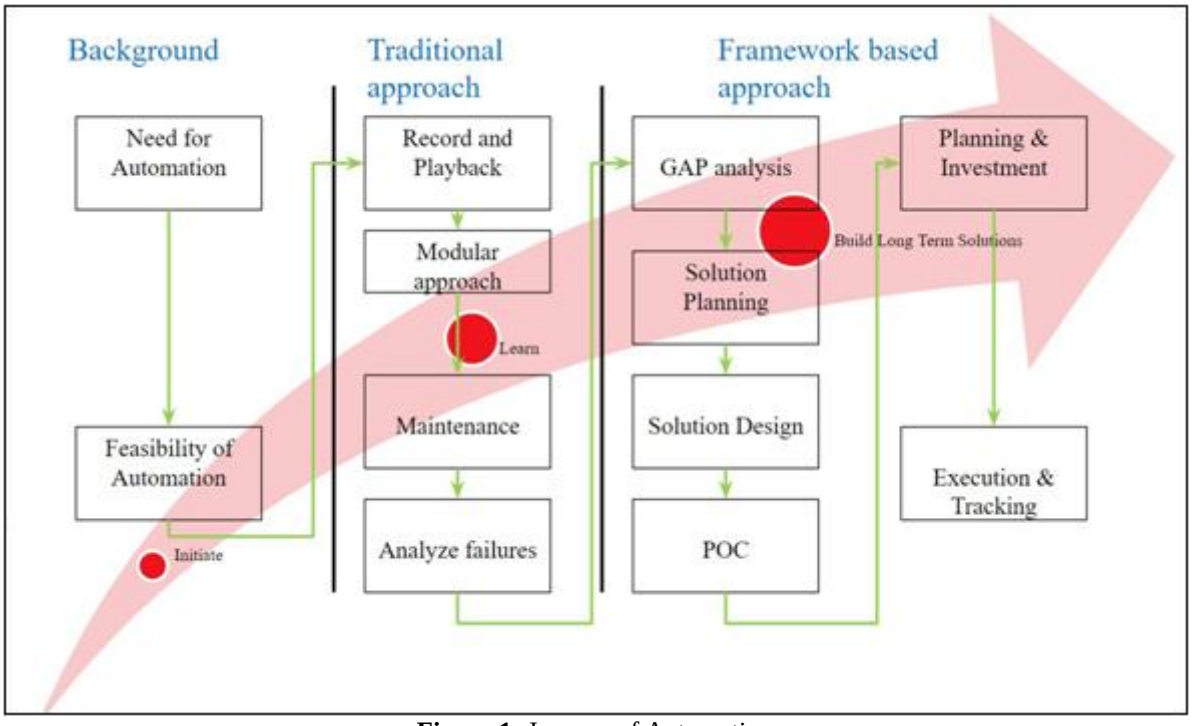

Figure 1: Journey of Automation

In summary, the journey towards automation is a strategic progression. It begins with evaluating the feasibility of automation, progresses into the manual and scripted phase, and ultimately transitions into a framework-based methodology to achieve efficient, scalable, and sustainable automation solutions (Hwang & Jung, 2010).

Through this expedition, establishments unlock the complete advantages of automation, empowering them to attain heightened productivity, improved precision, and diminished operational expenditures.

4. Traditional Approach

Certainly, provided below are explanations regarding the

drawbacks associated with maintenance, reusability, and performance in the traditional approach to automation:

1) Maintenance:

Maintenance involves the ongoing effort required to ensure the effective functionality of an automation framework or test suite. In the traditional approach to automation, several challenges pertaining to maintenancecan be observed:

a) Fragility: Test scripts may be tightly interconnected with the current state of the application, and even minor alterations in the user interface or functionality of the application can cause multiple test cases to fail. This necessitates constant updates to the scripts, which can be time-consuming and prone to errors.
b) Script Debugging: The process of debugging can be complex and time-consuming in traditional automation. When test scripts fail, identifying the root cause can be challenging, and rectifying the issues often involves sifting through code, which makes it difficult to quickly identify errors.
c) Version Control: Managing different versions of test scripts, particularly when working on multiple projects or applications, can become a cumbersome task in traditional automation. Keeping track of changes and ensuring the correct versions are utilized can be susceptible to human error.

2) Reusability:

Reusability is a crucial aspect of automation as it allows for the utilization of existing test scripts and components for various test scenarios. However, traditional automation has certain limitations when it comes toreusability:

a) Script Duplication: Test scripts are often created with a specific use case in mind, and they may not bedesigned for easy reuse. Consequently, teams tend to duplicate code or modify existing scripts to suit new scenarios, resulting in the proliferation of similar, but not identical, test cases.
b) Maintenance Impact: The process of reusing scripts with modifications can have an impact on the maintenance process. A change made in one place may inadvertently affect other scenarios where the script is used, leading to unintended consequences.
c) Limited Component Reuse: Reusable components, such as test libraries or functions, are not as prevalent in traditional automation. This restricts the ability to efficiently create and maintain a library of reusable building blocks for testing purposes.

3) Performance:

In automation, performance refers to the efficiency with which tests are executed, providing prompt feedbackwithout unnecessary delays. Traditional automation approaches may encounter the following performance challenges:

a) Execution Speed: Test scripts developed using the traditional approach may not be optimized for speed. They may include unnecessary waits, excessive interactions with the user interface, or redundant steps, all of which can slow down the execution of tests.
b) Scalability Issues: As the number of test cases increases, the scalability of traditional automation can become problematic. Managing large test suites, orchestrating parallel execution, and optimizing resource usage can pose challenges.
c) Resource Consumption: Traditional automation can be resource-intensive, consuming significant memory and processing power. This can impose limitations on the number of tests that can be executed concurrently on a given infrastructure.

To overcome these disadvantages, modern automation practices such as keyword-driven frameworks, data-driven testing, and behavior-driven development (BDD) focus on enhancing maintenance, reusability, and performance, thereby making automation more efficient and sustainable.

5. The Solution

The utilization of the Keyword-Driven Approach in the realm of test automation is a highly commendable solution when compared to conventional approaches due to several compelling reasons. (Pajunen et al., 2011) Particularly, this approach proves to be advantageous for individuals possessing extensive domain expertise but limited knowledge of the intricacies of automation frameworks. Herein lies the rationale behind this assertion:

1) Abstraction of Technical Aspects:

a) Traditional Approach: Within the realm of traditional automation, tests are frequently scripted using coding languages, necessitating a profound comprehension of the technical intricacies of the automation framework. Test cases become closely intertwined with the implementation specifics of the application under scrutiny, thereby posing challenges for domain experts in terms of test creation and maintenance.
b) Keyword-Driven Approach: Keyword-Driven testing abstracts the complexities of technical aspects. Test cases are scripted in a more user-friendly language, employing a set of keywords or commands thatprovide a description of test actions. This allows domain experts to focus on the overarching functionality and logic of the tests without the need for coding.

2) Segregation of Concerns:

a) Traditional Approach: In the realm of traditional automation, domain-specific logic, such as business rules, often becomes intertwined with automation code. This amalgamation of concerns can lead to confusion and difficulties in maintaining the test suite.
b) Keyword-Driven Approach: Keyword-Driven frameworks encourage a clear segregation of concerns. Domain experts have the ability to define test steps and their anticipated outcomes using keywords, while automation experts or testers handle the technical implementation of these keywords .This segregation simplifies collaboration and upkeep.

3) Components that are Reusable and Modular:

a) Traditional Approach: Traditional automation scripts often lack modularity, resulting in challenges when attempting to reuse components across different test cases. Alterations made in one script can inadvertently impact other test cases.
b) Keyword-Driven Approach: Keyword-Driven frameworks promote the development of reusable and

modular components. Domain experts can define custom keywords that represent actions or operations specific to their domain (Sai & Adline, 2017). These keywords can be reused across multiple test cases, thereby enhancing efficiency and consistency.

4) Accessibility for Domain Experts:

a) Traditional Approach: Domain experts possessing limited knowledge of automation may encounter difficulties when actively participating in the test automation process. Their contributions are often restricted to defining requirements or conducting manual testing.

b) Keyword-Driven Approach: Keyword-Driven frameworks empower domain experts to directly engage in the creation of test cases. They can define keywords based on their domain knowledge and collaborate with automation experts to implement these keywords as part of the framework.

5) Ease of Maintenance:

a) Traditional Approach: Traditional automation scripts can be delicate and require regular updates as the application evolves. Domain experts may encounter challenges when attempting to effectively maintain these scripts.

b) Keyword-Driven Approach: Maintenance is simplified within a keyword-driven framework. If the application undergoes changes, updates primarily need to be made at the keyword level, and domain experts can make these changes without delving into intricate code.

Table 1: Comparative analysis

Criteria	Traditional Approach	Framework Approach
Maintenance	High	Low
Performance	Low	High
Repository Size	Large	Small
Future Enhancement efforts	High	Low
Reusability	Low	High
Initial Investment	Low	High
Reliability	Low	High

The Keyword-Driven Approach is an accessible and efficient means of test automation that empowers domain experts to actively contribute to the creation of tests. By abstracting technical details, promoting segregation of concerns, and facilitating the development of reusable components, this approach provides a robust framework for collaboration and maintenance, making it an exceptional choice for domain experts possessing limited knowledge of automation.

Approach to Keyword Driven Framework

Implementing a Keyword-Driven Framework entails a series of pivotal stages, ranging from the initial planningand design to the subsequent execution and maintenance. In this section, we present a comprehensive approach to each of these stages:

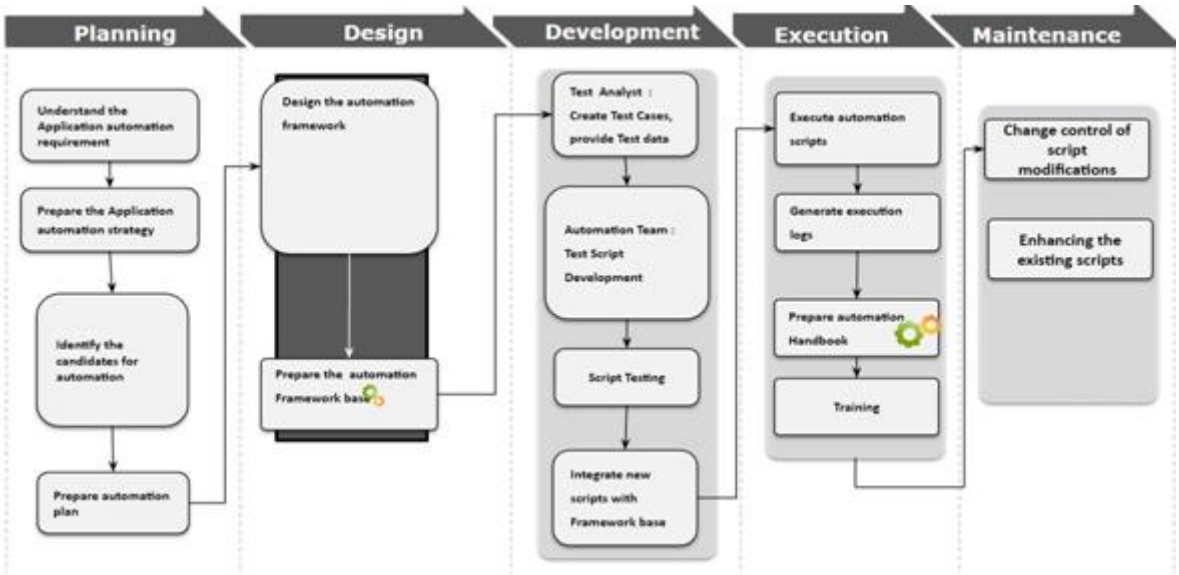

Figure 2: Keyword Driven Framework Implementation flow

1) Planning:

a) Understanding Requirements: It is crucial to gather intricate details regarding the application that is being tested. This encompasses a thorough understanding of the functionalities that need to be tested,the various test scenarios, as well as the automation objectives.

b) Automation Strategy: Facilitating effective collaboration between domain experts and automation

c) engineers is vital. Domain experts provide valuable insights into the behavior of the application and aid in the identification of keywords that accurately represent the actions performed within the application.

d) Tool and Candidate Selection: The choice of an appropriate test automation tool, candidate and framework that supports keyword-driven testing is of utmost importance. Considered options include Selenium, Robot Framework, or even custom-built frameworks.

e) Plan and Scope: Clearly defining the scope of automation is essential. This involves identifying the specific test cases, test scenarios, and application areas that are to be automated.

2) Design:

a) Design of Keywords: Collaborating with domain experts is necessary to accurately identify and document keywords. These keywords should effectively represent high-level actions or operations within the application.
b) Structure of Test Scripts: The structure of test scripts needs to be carefully designed. Each test script
c) should consist of a sequence of keywords, along with their corresponding parameters (Qian et al., 2013).
d) Reusable Components: Developing a library of reusable components that encapsulate the logic for executing keywords is highly advantageous. These components can be shared across multiple test scripts, promoting efficiency and reusability.

3) Development:

a) Development of Automation Components: The creation of the keyword execution engine is a crucial aspect of the development phase. This engine is responsible for interpreting and executing keywords, and should possess the capability to read test scripts, resolve keywords, and execute actions accordingly.
b) Authoring Test Scripts: Automation engineers and domain experts collaborate in authoring test scripts by selecting and arranging keywords in a logical sequence. Additionally, parameters for each keyword are specified.
c) Framework Configuration: Configuring the test automation framework to align with the keyword-driven approach is imperative. This may involve setting up the necessary libraries and plugins for the chosen tool.

4) Execution:

a) Execution of Test Cases: Test scripts are executed using the keyword-driven framework. The framework interprets the keywords, calls the corresponding automation components, and carries out the execution of the test cases.
b) Integration of Test Data: Test data is injected into test scripts based on the data-driven approach. This enables the execution of the same test script with different data sets.
c) Implementation of Logging and Reporting: Comprehensive logging and reporting mechanisms are implemented. These mechanisms capture details of the test execution and results, aiding in the identification of issues and debugging of problems.

5) Maintenance:

a) Regular Updates: It is imperative to regularly assess and revise the keyword library and framework in order to accommodate any alterations in the application that is being tested. It may be necessary to modify both the keywords and test scripts as the application progresses.
b) Keyword Maintenance: In the event of any changes occurring within the application, it is crucial to update the keywords accordingly so as to accurately reflect the functionality of the said application.
c) Regression Testing: It is vital to verify that the existing test scripts and keywords function correctly even after updates have been made. Regression testing should be conducted to detect any potential issues.
d) Scalability: It is essential to ensure that the framework possesses the capacity to adapt to new keywords and modifications in the application's functionality as time progresses.
e) Documentation: It is of utmost importance to keep the documentation pertaining to keywords, test scripts, and framework components up to date. This will facilitate the process of maintenance and encourage collaboration.

By adhering to this approach, one can effectively strategize, devise, develop, execute, and sustain a Keyword-Driven Framework that promotes efficient and manageable test automation. This approach empowers both domain experts and automation engineers to seamlessly collaborate and adapt to any changes that may arise within the application being tested.

6. Challenges

Implementing a Keyword-Driven Framework in test automation can yield a multitude of advantages, however, it also presents a set of specific difficulties. Here, we outline several common challenges that organizations may encounter during the implementation of a Keyword-Driven Framework:

1) *Keyword Identification and Maintenance:* Initial Identification: The process of defining and identifying the appropriate set of keywords that encompass all essential test scenarios can prove to be a formidable task. This endeavor necessitates close collaboration between domain experts and automation engineers (Yue-qin, 2009).
2) *Domain Expert Involvement:* Dependence on Domain Experts: The efficacy of a Keyword-Driven Framework is heavily contingent upon the availability of domain experts who can provide precise keywords. In the absence of readily accessible domain experts, the implementation process may encounter delays.
3) *Training and Familiarity:* Skill Set: Automation engineers and testers must undergo training in the utilization of the keyword-driven approach, particularly if they possess limited experience with this methodology. This training process can be time-consuming.
4) *Framework Development:* Initial Setup: Constructing the framework capable of interpreting and executing the keywords requires a significant level of exertion and expertise. It may entail substantial upfront development efforts.
5) *Keyword Library Maintenance:* Regular Updates: As the test application evolves, the keyword library and test scripts must be regularly updated. Maintenance can become intricate, particularly when dealing with a substantial number of keywords and test cases.

Despite these challenges, a well-implemented Keyword-Driven Framework can vastly enhance the efficiency and sustainability of test automation. Overcoming these challenges often necessitates meticulous planning, training, and continuous collaboration among team members and stakeholders.

7. Conclusion

The article delineates the difficulties associated with the lack of an automation framework and the dependence on conventional approaches, such as script-based automation. These difficulties encompass matters pertaining to maintenance, reusability, and performance, which highlight the limitations of traditional automation methods. As a remedy to these limitations, the article underscores the benefits of a Keyword-Driven Framework. This methodology enables subject matter experts to actively engage in test automation without extensive programming expertise. The keyword-driven approach provides a structured, reusable, and maintainable methodology for automated testing.

Ultimately, the article accentuates the significance of a strategic shift towards keyword-driven automation frameworks and logical analysis in framework selection. It elucidates how this transition optimizes testing processes, enhances efficiency, and empowers subject matter experts to assume a pivotal role in test automation. This strategic shift, guided by logical analysis, guarantees that organizations can make informed decisions in selecting the most suitable automation framework, thereby paving the way for improved software quality, enhanced productivity, and better overall outcomes in the ever-evolving realm of test automation.

References

[1] Bajpai, R. N. (2011). *A Keyword Driven Framework for Testing Web Applications. International Journal of Advanced Computer Science and Applications*. 10.14569/IJACSA.2012.030302
[2] Bhondokar, B., Ranawade, P., Jadhav, S., & Vibhute, M. (2013). *Generic Test Automation and Keyword Driven Approach*.
[3] De, H. (2009). *Research and Implementation of Keyword Driven Based on Robot Testing Framework*.
[4] Hwang, Y.-S., & Jung, S.-M. (2010). *A Keyword-based UI Test Framework for Web Services*.
[5] Pajunen, T., Takala, T., & Katara, M. (2011). *Model-Based Testing with a General PurposeKeyword-Driven Test Automation Framework*. 10.1109/ICSTW.2011.39
[6] Qian, Z., Zhe, J., & Zeng, Z. (2013). Keyword Driven Automation Test. Applied Mechanics and Materials. 10.4028/WWW.SCIENTIFIC.NET/AMM.427-429.652
[7] Yue-qin, Q. (2009). *Research of Keywords Decomposing Method in Keyword-Driven Framework.*