Automation Framework Development - Best Practices
Abstract

These day’s systems are complex with rich User Interface, backend and that too require multiple capabilities. Test Automation Framework are complex to develop and maintain. This white paper gives an idea about the implementation of single framework for front-end and back-end which covers all aspects of testing.

The major purpose of this document is to define a common set of Automation development guidelines and how the Selenium with Java can be used to develop a single automation framework.

This white paper looks at applying the Hybrid (KDF and Data Driven) framework approach for automating the test design. The KDF approach not only saves effort and cost but increases the usability of framework across the different teams and reduces the dependency on the resources with automation skills.

This paper sets out the steps involved in building the automation framework.

Introduction

Of total effort spend in an application development, at least 1/4th effort goes towards testing. Of this testing effort, 70% go towards functional testing. Automating functional test cases involves developing test scripts using scripting languages like python, Perl or any programming language like java. Companies are constantly looking for the ways to develop an automation framework which can be used across the teams and resources with no scripting knowledge.

Keyword Driven Framework approach offers an elegant way to tackle this challenge.

Test automation refers to the development and usage of tools to determine the success or failure of pre-specified test cases, against the Application Under Test, without human input.

This white paper attempts to outline the selection of appropriate tool and designing the test automation framework.
The Urge for Automation Testing

In the initial stages of an application development, as more features are added, manual exploratory testing becomes crucial. As a product matures, the complexity of the app increases.

Relying solely on manual testing to execute repetitive testing steps is not only costly but also time-consuming and ineffective. Testing with the use of tools or automation testing is the best solution when you have a lot of regression cases.

For better accuracy and speed, test automation is vital to cover all corners of the application. A combination of manual testing and test automation can give your software product maximum test coverage. Inadequate coverage can lead to incorrect feedback and increase the risks of costly software errors.

Problems in Automation Testing

Developing automation scripts for test automation requires fancied scripting skills. Someone professional in scripting would rather be a developer than a tester. The availability of test automation engineers with a developer mindset trained in one of the languages used for test automation scripts is a challenge.

In linear Test automation, reusability of automated scripts across different products or different set of data is difficult. Cross Browser testing for a web application requires different scripts for each browser in linear test automation approach.

In the absence of any test automation frameworks, the identification of the page objects will be arbitrary, it could be record on the current GUI layout for an instance. In a future version of the application, these objects could change – say, due to the addition of other links. In this case, each test script involving an application screen that has this set of links will be broken, and will need rectification. This requires a significant rework effort, as there could be many such screens and perhaps the whole web application.
Test Automation Frameworks

To overcome the challenges in linear automation testing, there arises a need of Automation Frameworks.

Test Automation Framework provides the Object-oriented approach and eliminates the hard coding.

Based on our experience, test automation framework provides the re-usability of automated scripts for a different product under test. However originally those scripts were created for automation of different product.
Test Automation Frameworks

Frameworks which are widely used can be classified into three categories.

- **Data Driven**: This framework is used when user want to test the same requirement multiple times with different sets of input data. In this framework data is stored externally, which can be CSV files, excel sheet, xml files or text files.
  - To perform the same action with different sets of data, we follow the Data Driven approach
  - Data sets are provided in an excel sheet and in the test case we pass the reference of data excel with cell range

- **Keyword driven**: This framework is also known as Table-Driven testing or Action Word based testing framework. In this framework, each test step is performed based on the action called.
  - Automated Test Scenarios are the excel sheets instead of typical scripts.
  - Automation Engineers should select the keyword from the list based on the action to be performed.
  - Pre-defined subroutines associated with actions
Hybrid: This framework is combination of Data driven and keyword driven framework. It is the very fast way of creating automation and requires very less programming experience, once automation framework is ready.

**Benefits of Automation Framework**

**Saving of Manual Testers’ time**
The saving in a test engineer’s time is almost linearly proportional to the number of test cases automated, and the number of test cycles to be executed.

**Repeatability**
In complex test environments (those involving several application components, platforms, environment variables, etc.) human error can creep into manual test execution. Automation ensures 100% repeatability and hence greater predictability in execution.

**Easy to Maintain**
Modular approach is always easy to maintain than the linear scripting. Any enhancements to the test scripts are easy to implement in Automation Framework with modular approach.

**Enabling non-functional testing – synergy with other quality tools**
With additional tools and effort, it is often possible to configure special runs of the automated test cases in order to perform non-functional testing, for example:

- Performance testing
Scalability/ load testing
Memory profiling
Code coverage and impact analysis

Depending on the project’s priorities, the above benefits can be translated into higher quality, lower costs or lesser time to market.

**ROI on Automation Framework**

Based on the benefits outlined above, we recommend that automation should be seen as an overall quality and productivity improvement initiative, rather than merely as a cost saving exercise.

The monetary values of the following can be compared:

- **Return**: Saving of manual testers’ time, converted to a monetary value
- **Investment**
  - Cost of the automation tool
  - Cost of the automation effort

For automation of functional test cases, a typical ROI profile might look like the one below

**Key Highlights**

- **Increase in Test Coverage (Manual v/s Automation)**
  - 50% reduction in the time taken for a verification cycle and 5 folds increase in the number of test cases executed per cycle
- **35% Reduction** in the defects trends seen from the production releases
- **Break-even** for automation investment achieved in 7 regression cycles
Test Automation Life Cycle vs Application Development

It is very important to consider the Test Automation development as a separate project all together from the Application development. It is vital to synchronise the test automation life cycle with the application development life cycle.

This is depicted in the figure below:

Automation Testing Life Cycle

Automation Tools Selection

Based on the system under test and testing requirements, we need to identify the best tool for automation. Following four major criteria are suggested for selection of testing tools:

- Meeting requirements
  - Test tools should provide backward or forward compatibility with the application under test
  - Test tools should provide the required amount of trouble-shooting/debug/error messages to help in analysis.
  - The test tool must have some intelligence to proactively find out the changes that happened in the product and accordingly analyze the results

- Technology expectations
Test tools should allow test developers to extend/modify the functionality of the framework. Extensibility and customization are important expectations of a test tool.

Test tools should be cross platform. The scripts develop on some test platform should be compatible with other OS platforms.

- **Availability of required Skills**
  - Test tools expect the users to learn new language/scripts and may not use standard languages/scripts. This increases skill requirements for automation and increases the need for a learning curve inside the organization.

- **Management aspects**
  - A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already-expensive test tool. When selecting the test tool, it is important to note the system requirements and the cost involved in upgrading the software and hardware needs to be included with the cost of the tool.

Following figures depicts how the appropriate tool is selected based on the factors mentioned above.
What to Automate, and When

We do not recommend making automation decisions solely based on (costs only) ROI. However, given that automation needs a non-trivial effort, it is advisable to undertake automation only after some parts of the application under test have become stable. This ensures that the automated test cases will get executed enough number of times to offset the automation effort. A few consequences of this rationale are as follows:

- Automation is much more attractive in product development, as compared to services projects.
- In product development, it would be prudent to start automation after a couple of releases of the product. This ensures that chances of a GUI overhaul are minimal.
- It is advisable to first automate those test cases that involve functionalities that are stable. New features could typically be taken up for automation in the next release. This also means that automation is most effective for regression testing.

Challenges in Automation

- The choice of the tool is often restricted by the technology underlying the application under test.
- In web applications, multi-window test cases are usually difficult to automate. Pop-ups, single child windows are not a problem.
- Integrations between applications under test are sometimes difficult to automate.
Automation Best Practices

We recommend the following best practices for automation.

✓ Since automation frameworks are essentially about abstraction, an important set of best practices deals with ensuring loose coupling between –
  ➢ The test data and the test scripts,
  ➢ Test scripts themselves,
  ➢ The automation framework and the application under test.
  ➢ The test cases and the automation framework, and
  ➢ The automation framework and the automation tool.

✓ Hybrid Framework (Keyword + Data Driven approach) should be developed to provide less scripting efforts for the manual engineers.

✓ Keyword names should be carefully chosen, so that human readability is also high. This enables gradual transitioning from manual testing to automated testing.

✓ Verification points should be judiciously inserted into the scripts. In case of test case failure, these points accelerate the process of zeroing in on the reason of the failure

✓ Framework should be integrated with the Test Management tool

✓ Generic Libraries should be developed to reuse the modules in different keywords.

✓ The development of an automation framework is similar to the development of an application in several respects, and hence should be planned and tracked as a (sub) project in itself. It should be noted that framework creation and test case design are distinct activities (and require different skills).

✓ Simpler test cases should be automated before complex ones. This makes it easy for later scripts to build on earlier ones.

Reasons for terming it as a best practice

Following are key reasons:

✓ **Script Less Framework**: Once keywords are developed, manual testers or developers can easily write test scripts for automation. No Scripting skill (PERL/PYTHON) required to create the test case excel sheet. Completely removed the dependency of scripting knowledge to automate

✓ **Aligned with business goal**: 100% aligned with customer business goal of successful transformation to AGILE from waterfall model. Enables each Scrum Member to automation quickly.
  o Easy to use across the different teams(Automation/Manual)
  o Dependency on team competency for a specific tool/language.
o Modular & Re-usable Framework
o Customized Reporting

✓ **Less time to Automate:** More than 80% reduction in automation scripting. User only need to select sequence of keywords in a spreadsheet rather than writing PERL/QTP/Java code.

✓ **Easy To Understand:** As it is maintained in Excel sheet and no coding is exposed, the test scripts are easy to read and understand. Keywords & actions resemble so closely manual test cases, which become easier to write and maintain.

✓ **Early Start:** You can start building Keyword Driven test cases before the application is delivered, as Object Repository can be easily set up at the later stage. Using information gathered from Requirements/documentation, keyword data tables can be created from manual test procedures.

✓ **Re-usability of component:** With implementing modularization in Keyword Driven, once written keywords are 100% reusable.

✓ **Highly Effective Reporting & hence Easy troubleshooting**

✓ **Automation Tool Independence:** Can be followed with any automation tools like QTP, Selenium etc.

---

**Summary**

Test automation offers a promising way of quality and productivity improvement in software testing. While manual testing is required and desired (except perhaps for a product that is purely in sustenance mode), the time and cost required for it can be significantly reduced. Moreover, a part of this saving can be invested for better quality. Commercial/Open source tools and a rapidly growing body of knowledge have led to a reduction in the time needed for monetary returns to be seen, thus accelerating the adoption of test automation in the industry.

We recommend that for all product development, at least the study phase of the test automation lifecycle should be undertaken.

---

**Key Takeaways for Audience**

✓ Innovations pay out the results in longer run.
✓ Reduction of automation efforts by 80% and hence increasing the coverage of automation features in same time
✓ Backbone of Agile methodology to enable quick automation in each Scrum team
✓ Even Manual Testers can do automation without having automation scripting knowledge & can do gradual ramp-up of automation skills without impacting the project deliverables
✓ Automation Left Shift by implementing the framework in UT cycles
✓ High reusability of one-time developed keywords across teams & features
✓ Independence of KDF Spreadsheets to Test Automation tools (QTP/Selenium) & Underlying Technology (HTML5/Flex )

References
✓ www.google.co.in
✓ www.softwaretestinghelp.com
✓ http://www.impactqa.in
✓ https://en.wikipedia.org

Author Biography

Umair Siddiqui has 7+ years’ experience of working on Automation Framework and manual testing of medium to large sized projects in Telecom domain. He is currently working as a Technical Leader in Aricent Technologies. He is awarded with Aricent Individual Excellence annual award. He has strong exposure of various software development life cycle including Requirement Understanding, Design, Feature Analysis, Test Strategy, Test Planning, Test Execution and Test Automation of assigned features

Girish Chandra has 7 + years of testing experience which includes automation and manual . Girish is Quality Assurance and Software testing specialist , managing testing efforts , developing test methodologies and creating standard procedures for quality assurance and testing .He has worked for different telecom technologies like 2G , 3G(UMTS),4G (LTE) and SIP .And has good experience on real time protocol testing .
THANK YOU!