Automation: Deep Dive Into Web, Mobile & API – Part 1

Muhammed Suhail TS[*]

[*] Neudesic, an IBM Company
[*] Senior Automation Consultant, Quality Engineering

DOI: 10.29322/IJSRP.12.12.2022.p13233
http://dx.doi.org/10.29322/IJSRP.12.12.2022.p13233

Paper Received Date: 5th November 2022
Paper Acceptance Date: 5th December 2022
Paper Publication Date: 20th December 2022

Abstract- Testing is a phase in software application development lifecycle. This process validates whether the application which is developed conforms to the requirements provided. This process could be accomplished using both manual as well as code based automated solutions. This paper mainly talks about the different types of test automation approaches which could be implemented to enhance the productivity and efficiency of the testing life cycle. The different automation strategies, generic framework architecture, the features and advantages and how returns are generated on the investment is discussed in detail.

Index Terms- SDLC – Software Development Life Cycle, AST – Automated Software testing, MT – Manual Testing, API – Application Programming Interface, IDE – Integrated Development Environment, POM – Page Object Model, E2E – End to End Tests, OTP – One Time Password, UI – User Interface, CI/CD – Continuous Integration/Continuous Delivery (Deployment), ROI – Return On Investment, VCS – Version Control System

I. Introduction

Any process which happens n number of times consistently with only change in data are known as static processes. An intelligent decision making is not required for such type processes. Any process which has this kind of property is an ideal candidate for Automation. Automation is the process by which such repeatable steps could be made to do by machines or by software code. This results in the process becoming faster, highly efficient, reliable and error free. This drastically reduces the human intervention required for the process to be completed. Automation techniques can be broadly classified into two viz. hardware-based automation and software-based automation.

IT software testing refers to analyzing and interrogating a software program to unearth errors and secret faults. Software testing is projected to ensure that an integrated software program performs to meet satisfaction thresholds, access, and quality preservation. For an Information Technology system to be complete, the software testing realm should not miss in the Software Development Life Cycle. Under the requirements of SDLC, software development is not absolute under it is subjected to the testing process. Inherently, testing is not performed to demonstrate an error-free system but to establish a confidence wall that supports the installation and performance of the entire system.

In this research paper, the major discussion will be on the software automation practices. Software testing is expensive, labour intensive and consumes lot of time in a software development life cycle. There was always a need in software testing to decrease the testing time. This also resulted to focus on Automated Software Testing, because using automated testing, with specific tools, this effort can be dramatically reduced and the costs related with testing can decrease. Manual Testing requires lot of effort and hard work, if we measure in terms of person per month. Automated Software testing helps to decrease the work load by giving some testing tasks to the computers.

In software automation software code is created, which helps to configure and run repeatable processes, n times consistently. Key points of discussion will revolve around software testing area and how automation practices helps in performing automated software testing effectively. In software testing, Test Automation is the use of software separate from the software being tested to control the execution of tests and the comparison of actual outcomes with predicted outcomes. The testing process mostly adheres to the static process category wherein there exists pre-defined input steps and corresponding outputs steps. Here a known outcome of a particular action is presented which is in turn validated via code.

II. Types Of Test Automation

Test automation could be employed in software testing on the below types of applications:

1. Web based application automation
2. Mobile based application automation
3. API automation

Web automation includes automation testing an application which is hosted over web on a particular URL and validating and verifying its conformity to the requirements outlined during the application development. The web automation is done from a UI standpoint and tests are run which mimics the action of a manual user via code. The applications developed caters to various fields like travel, tourism, banking, hospitals, insurance etc.

Mobile automation is similar to that of web, but instead of the application being hosted on a URL it is developed as a distributable installer which is installed and accessed over multiple type of mobile devices. The automation testing of this is again accomplished via UI components and it also mimics the action that a manual user would perform on the device via code.

API automation is different from the above two approach. API's are the internal connections which are present on both web and mobile application, which is hidden from the user, but critical for the application to function. It is analogous to human body containing blood vessels and nervous system. Hence when performing automation in this layer it requires specific targeted information, inputs and outputs.

Advantages of Automation

Test Automation is highly advantageous as compared to MT. That is the reason why it is highly adopted. Some of the key advantages of test automation are given below.

1. Enhanced Results

Since automation testing saves plenty of time even when complex and enormous systems are taken into consideration. This allows testing to be carried out repeatedly, delivering better and faster results with significantly lesser efforts and reduced time.

2. Swifter Feedback System

Automation testing is extremely crucial during the validation phase of any software project. It significantly enhances communication among the developers, designers, and product merchants, and provides space for the potential glitches to be rectified immediately thus enhancing the efficiency of the development team. The feedback loop is extremely faster.

3. Brand Enhancement

The effectiveness of testing is always dependent on the quality of test data that is being used. Testing is often performed on the copies of live databases as creating relevant and quality test data takes copious amounts of time. Automation solutions allow you to re-use your data time and again. This saves a lot of costs from project handling and project maintenance perspective.

The best aspect of automated testing is that it adds value to all the stakeholders. Automated testing systems not only enhance the system's capability but also pave the way towards digital innovation and revolution. It not only improves the brand name but also increases brand recall value, thus ensuring far greater customer retention. Due to automation testing, there are permanent fixes generated to issues long pertained as unsolvable.

4. Cost-effective

Even though the initial investment needed for automation testing is on the higher end, it saves a lot of money for the company in the longer run. It is predominantly due to the reduction in the amount of time required to run the tests. It also contributes a much higher quality of work as there are no chances of neglect or human error. This decreases the necessity of fixing glitches in the post-release phase, thus saving huge amounts of project costs. Additionally, the number of runs is also one of the factors that results in returns over investment,

5. Efficiency testing

Testing is one of the most pivotal parts of the entire application development cycle. The most attractive part of the automation testing is that it can be left virtually unattended. This leaves a lot of room for the results to be monitored towards the end of the process. Tests could be triggered and run-on nightly basis. This allows for increasing the overall efficiency of the application.

6. Increase in coverage area

Through the use of automation testing, more tests can be allotted pertaining to any application. This leads to higher testing coverage and a reduction in software anomalies. It also allows room for testing more features and complex applications. By covering the smoke and regression tests via automation quality time could be invested in ad hoc testing. However, in order to do the same thing in a manual testing scenario would require a massive team along with heavy time constraints.

7. Detailed testing

All testers tend to have different testing approaches with different focus areas as per their exposure and level of expertise. With the help of automation, there is an equal focus on all areas of testing, thus assuring the best possible quality of the end product with greater emphasis on each aspect of the product. Automation testing is known for its atom level approach of testing due to which it is considered robust & error-free.

8. Reusability

Test Automation is repetitive in nature due to the nature of its test automation cases. It is easy to setup, configure and gives the software developers an opportunity to assess the program's reaction. Automated test cases are totally reusable and hence can be utilized for testing any aspect of the code as per significance and through a plethora of different approaches. Framework based models helps to achieve this with ease.

9. Earlier detection of defects

Automated test runs are triggered instantly when the development code is updated on the main branch. Automation testing documents the software defects and hence making it considerably easier for the testing teams. This also makes it relatively easier for the development and support team to together contemplate the defects and give a faster output. The overall development speed of the project is increased while ensuring correct functionality across relevant areas. The earlier any defect is identified, the better and cost-efficient it is to solve and deploy it.

10. Time to market

Test automation helps significantly in reducing the time-to-market launch of an application. Automation testing allows constant and regular execution of test cases. Post automation the test library execution is extremely swift and runs longer. Environment wise releases are triggered on successful completion of each stage. This results in faster production deployments.

III. Web Automation

Web automation testing technique is where manually written test cases are converted to have a corresponding code and the code gets validated against the application for the same steps as in manual mode.

Consider the example below. The below given is a generic scenario in web application test.

Scenario Description	Steps	Expected Result
Login	1. Launch the application 2. Navigate to the home page 3. Enter valid user id and password 4. Click on Login	The user should be successfully logged in to the system

During automation for each of the steps defined will have a corresponding code snippet which performs that operation and finally a validation piece of code, which validated the expected result.
There are many open-source as well as paid tools which could be leveraged for Web Automation purposes. The below table (Fig 1. Web Automation Tools) lists a set of tools that are available in market, which is used for Web Automation. The parameters under consideration during the evaluation of the software's are License Cost, Application Type, Record & Playback, Programming Language support, Platform Support, Browser Support, Technical Support, Dialog box Support, Creation of Scripts, Usage, Data Driven Framework & Report Generation.

Features	Selenium	Quick Test Professional	Test Complete
License Cost	Open source	Licensed. Very expensive, costs about 13000$	Licensed. Costs about 1999$
Application Type	Web applications only	Web and Mobile applications	Any type of application (web, desktop)
Record Playback	Support	Support	Support
Programming Language Support	Java .Net Perl PHP Python Ruby	VBScript JavaScript	VBScript JavaScript Delphi Script C++ C#
Platform Support	Windows MAC UNIX LINUX	Windows only	Windows only
Browser Support	All browsers	IE Firefox Chrome	All browsers
Technical Support	No official technical support	Good technical support	Good technical support
DialogBox Support	Partially supports	Supports all major kinds of dialog boxes	N/A
Creation Of Scripts	Not powerful as many actions are not recorded by the IDE	Powerful to some extent than selenium	N/A
Usage	Need experience and programming skills	Easy to learn/use/edit/ parameterize/playback the VB script	Need experience
Data-Driven Framework	Excel- CSV	Excel files Text files XML DB files	CSV Excel SQL
Report Generation	HTML	HTML	HTML, XML

Fig 1. Web Automation tools

IV. Mobile Automation

In the modern world majority of the handheld mobile device market is ruled by two Operating System (OS), namely iOS & Android. Android is used globally by around 71.62% of devices whereas iOS contributes 27.73%. On the remaining 0.6% is occupied by other vendors. Hence when mobile automation is considered it is mainly referring the automation of devices using iOS and Android OS. There are three types of applications that runs on mobile devices:

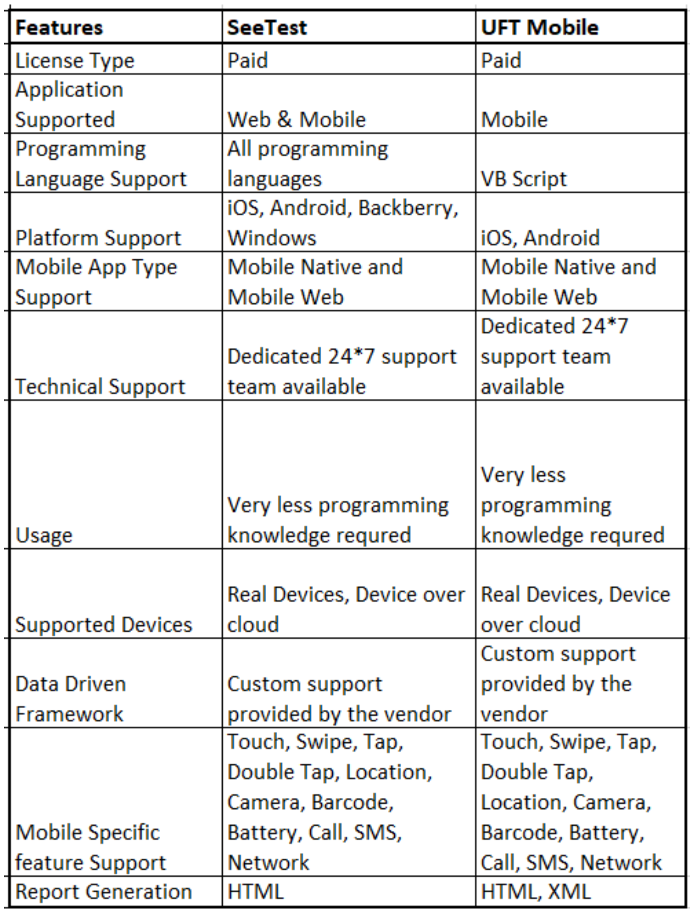
a. Native Applications
b. Browser based Applications
c. Hybrid Applications

Native applications are applications which are coded and developed in the language which is supported by the OS. i.e, for iOS applications developed using swift language in Xcode IDE and which has a file extension of .ipa. In Android it uses either Android Java or Kotlin and the applications are developed in Android Studio which are having the file extension as .apk.

Browser based applications are developed to be consumed via mobile browsers. These are hosted on URLs and accessed via mobile browsers. For example, applications like amazon.com, booking.com etc. Automation of test cases on these applications can also be accomplished via mobile automation tools.
Hybrid applications are developed for both Android and iOS platforms using a single programming language. Ionic is a framework which leverages JS libraries such as Angular, React and Vue to build mobile applications that can be used by iOS and Android platforms together. In this approach the core development language for a specific platform is not used, rather a different programming language is used altogether.
The below given table (Fig 2,3. Mobile Testing Tools) contains a list of Mobile Automation tools and its capabilities listed. The major criterions included are License Type, Application Supported, Programming Language support, Platform Support, Mobile App Type Support, Technical Support, Usage, Supported Devices, Data Driven Framework, Mobile Specific Feature Support & Report Generation.

Features	Appium	Calabash
License Type	Open Source	Open Source
Application Supported	Mobile	Mobile
Programming Language Support	Java, Objective C, C#, Python, Ruby etc	Gherking, Ruby, Cucumber
Platform Support	iOS, Android	iOS, Android
Mobile App Type Support	Mobile Native and Mobile Web	Mobile Native only
Technical Support	Huge Community Support available	Limited Community Support
Usage	Requires intermediate level technical programming knowledge to start, setup and use the framerwork	Requires inetrmediate technical knowledge to setup and start using the framework
Supported Devices	Real Device, Emulators, Device over cloud	Real Device, Emulators, Device over cloud
Data Driven Framework	Excel, CSV	Cucumber examples, Excel, CSV
Mobile Specific feature Support	Touch, Swipe, Tap, Double Tap, Location, Camera, Barcode	Touch, Swipe, Tap, Double Tap, Location, Camera, Barcode
Report Generation	HTML	HTML

Fig 2. Mobile Testing Tools

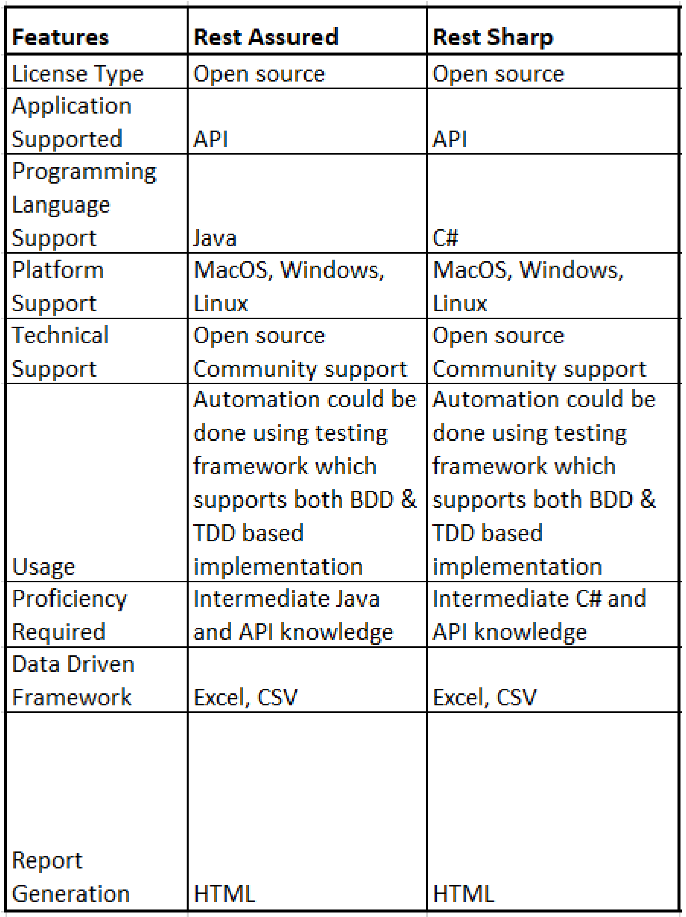


Features	SeeTest	UFT Mobile
License Type	Paid	Paid
Application Supported	Web & Mobile	Mobile
Programming Language Support	All programming languages	VB Script
Platform Support	iOS, Android, Backberry, Windows	iOS, Android
Mobile App Type Support	Mobile Native and Mobile Web	Mobile Native and Mobile Web
Technical Support	Dedicated 24*7 support team available	Dedicated 24*7 support team available
Usage	Very less programming knowledge requred	Very less programming knowledge requred
Supported Devices	Real Devices, Device over cloud	Real Devices, Device over cloud
Data Driven Framework	Custom support provided by the vendor	Custom support provided by the vendor
Mobile Specific feature Support	Touch, Swipe, Tap, Double Tap, Location, Camera, Barcode, Battery, Call, SMS, Network	Touch, Swipe, Tap, Double Tap, Location, Camera, Barcode, Battery, Call, SMS, Network
Report Generation	HTML	HTML, XML

Fig 3. Mobile Testing Tools

V. API Automation

API Automation involves the test automation of the invisible components present in every application built today. These components are the core part of the system which makes it functional, easy to use and helps in effective data handling. API is the middle layer of the application. For example, when a user registers the first time when using an application, this whole data is stored on the backend and is later retrieved when a specific condition/request invokes it. This process is handled by APIs. Automating the APIs of an application is really helpful as it gives very fast feedback and generally acts as pre requisite step before any UI based tests are triggered. That is, if an API test fails then there will definitely be an issue present in the UI of the application as all UI based tests basically invokes an API in the backend. Hence API Automation is highly effective in identifying issues in the system in a fast-paced manner.
The below given table (Fig 4,5. API Automation Testing Tools) contains a list of API Automation tools and its capabilities listed. The major criterions included are License Type, Application Supported, Programming Language support, Platform Support, Technical Support, Usage, Proficiency Required, Data Driven Framework & Report Generation.



Features	Rest Assured	Rest Sharp
License Type	Open source	Open source
Application Supported	API	API
Programming Language Support	Java	C#
Platform Support	MacOS, Windows, Linux	MacOS, Windows, Linux
Technical Support	Open source Community support	Open source Community support
Usage	Automation could be done using testing framework which supports both BDD & TDD based implementation	Automation could be done using testing framework which supports both BDD & TDD based implementation
Proficiency Required	Intermediate Java and API knowledge	Intermediate C# and API knowledge
Data Driven Framework	Excel, CSV	Excel, CSV
Report Generation	HTML	HTML

Fig 4. API Automation Testing Tools

Features	Postman	SOAP UI
License Type	Paid	Paid
Application Supported	API	API
Programming Language Support	Javascript, Python	Groovy, Javascript
Platform Support	MacOS, Windows, Linux	MacOS, Windows, Linux
Technical Support	Dedicated 24*7 support team	Dedicated 24*7 support team
Usage	Evaluation and custom checks are written on the test section provided for conditional validation	Evaluation and custom checks are written on the test section provided for conditional validation
Proficiency Required	Basic Javascript knowledge	Basic Javascript knowledge required
Data Driven Framework	Excel, CSV, Json, XML	Excel, CSV, Json, XML
Report Generation	Custom report provided by the tool which can be exported different options like csv, json etc	Custom report provided by the tool which can be exported different options like csv, json etc

Fig 5. API Automation Testing Tools

VI. Automation Framework Generic Architecture

For any type of test automation, it is always better to build it via a framework. Frameworks basically helps to handle different modules as individual components operating independently. This makes the code more readable, robust, re-usable and easy to maintain. In addition to this as the number of test cases grow, it would become impossible to maintain the code if it is not properly structured. There are multiple types of architecture which are available for creating a test automation framework. The most widely used and common one is referred to as the Page Object Model (POM).
In POM Model separate classes are created for each specific pages of an application. Each of these page classes will contain the elements which are present on the corresponding web page and its corresponding actions functions that helps to perform some activities on the page. For example, on a login page of an application there will be two text field, viz. User name, Password and a button Login which when clicked would validate the data entered and allow the user to login. Hence on the login page class of the code that is written, element identification for the three fields will be present. Additionally, it would contain the enter text actions for User name, Password and a click action for Login button. These would leverage functions which are written on a common class accessible by all the page classes for click and enter text with only difference in data. The below given architecture diagram (Fig 6. Generic POM Architecture) explains how a basic page object framework works.

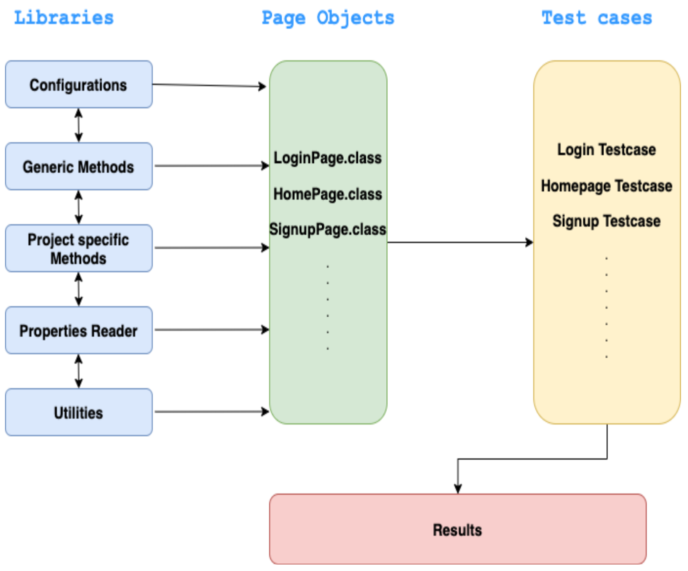


Fig 6. Generic POM Architecture

In the above diagram, the utilities libraries, including the core of the project includes and two-way binding which helps to access the methods and functions written in those libraries between themselves and the page object classes. The page object classes then leverage all the required reusable functions present in those libraries and the test case then call the custom functions created on the page classes to accomplish the expected steps and conditions defined on the test cases written. On top of the test cases a testing framework is also implemented on the framework to order and run the tests in an effective manner. Finally, once the execution gets completed results are generated which includes, passed, failed and skipped test cases. The report could have details like screenshots, logs added, which is really useful when there is a requirement to debug the test failures.

VII. The Automation Triangle

The Automation Triangle is a pyramid which, if followed makes test automation more efficient and effective. It helps to visualize the steps to be taken to tackle automation testing. The below figure shows the automation triangle (Fig 7. Automation Triangle).

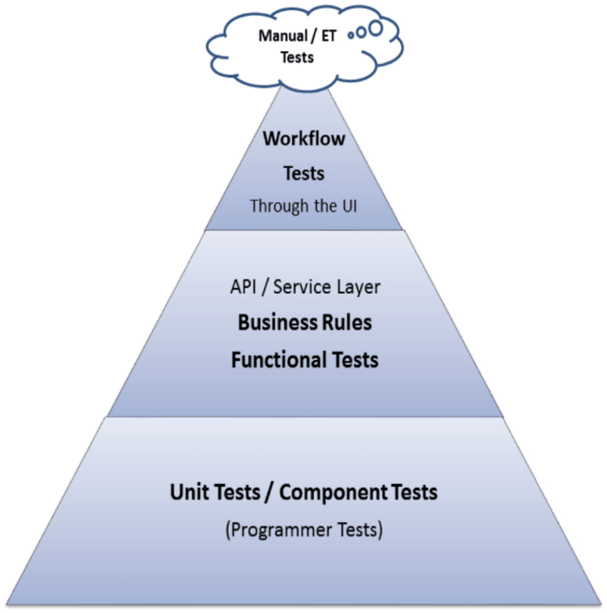


Fig 7. Automation Triangle

The lowest layer of this pyramid is occupied by the Unit Testing piece. This is most effective in covering many of the standard error scenarios which are present inside a piece of code. The unit tests are created by the developers to cover the boundary and basic positive and negative scenarios, present in the code that they develop. Unit tests are quick to write, assure internal code correctness, and can be run in minutes before committing new code changes. These tests provide the solid foundation for automation testing. However, these tests are not good enough to cover all aspects of a software.
The next layer where automation needs to be implemented is the API layer. The API layer contains core functional and business rules which enables the application to function properly. It is highly important to validate the rigidity and accuracy of the APIs, as these are the core components of the system. Hence providing automation in this layer helps in enhancing the correctness and reliability of the application. These tests may take more time to create and, if they access databases and other services, provide slower feedback. If business stakeholders need to collaborate on acceptance tests to verify business rules, these middle layer tests can be written in plain language that everyone can understand.
Now the topmost layer for automation is the workflow through the UI. UI automation tests written for this layer helps in reducing the functional testing overhead. In order to validate basic UI flows in an application having UI it is best to have automation take care of this. On simple applications, the number of UI tests might be lower, likely between 50-100 tests. Whereas on complex application there exists >1000 tests from UI that needs to be validated on every release. By providing automation in this layer, it helps reduce the overall testing effort considerably especially when the product has multiple release cycles.
The little cloud at the top of the pyramid represents testing that's done manually. Even with solid automation at all layers of the triangle, the product might require a huge cumulonimbus cloud that includes exploratory testing, accessibility testing, security testing and many other critical testing activities. Automation will help to speed up this manual testing process.

Prerequisites for Automation

There are multiple types of test suites which are considered as candidates for automation. The major three types of such test suits

1. Smoke tests
2. Regression tests
3. End to End tests (E2E)

Smoke test suite basically contains the least number of tests, which ensures that the application is functional. It is executed as the first type of tests on a new release build provided for an application. Since the number of tests are minimal, it runs the fastest.
Regression tests are those which requires a huge time to complete as it will cover all the existing as well as newly created functional components of an application. Regression suites typically contain the greatest number of test cases.
E2E tests includes the complete end to end flow of an application. Suppose the application is an e commerce application where suppliers list their products and the users buy from them. Here, one E2E script will start with the supplier adding the item on the platform till the buyer checks out and receives the item. This covers the flow of the application in its entirety.
There are certain things which should be obtained when starting with test automation in any project. Those are listed below:

- Ensure that proper test cases are available in the Project Management tool that could be leveraged for automation.
- Proper mapping of test cases with requirements is mandatory.
- Review each test case before automating to ensure ambiguous steps are not present.
- Identify a proper sample of tests, which could be first worked on as a Proof of Concept (POC) before the actual automation starts.
- Ensure that efforts including test scripting, debugging and review is considered during the estimation phase.
- High level strategy should be prepared to be followed during the entire phase of automation.
- Some test cases like OTP based on mobile, should be considered as non-automatable initially because the automation of these process would be highly time consuming and would reduce the overall productivity of the automation process. Hence such test cases should be taken as last.
- Consider the smoke suite for the initial phase of automation, as these would most reused and hence provide ample reduction in efforts.
- If APIs are present then automate the API layer first as it is faster to run and provided results in a fast pace manner. Also, API automation is more reliable than UI.
- Ensure that automation tests are having a proper architecture and a framework that supports it.

VIII. ROI STRATEGY

Most of the time it is thought that automation is a magic which would bring huge savings for an organization, if implemented. This would work only if test automation is implemented with a really good strategy.

For implementing test automation in a project, a strategy needs to devised first. Mostly the smoke tests are taken for initial implementation followed by regression tests. Only after these cases are covered, in sprint automation tests should be considered in an agile project. In sprint should be taken in an N-1 sprint approach, i.e., if the current sprint is Sprint 10, then automation test cases would be of Sprint 10-1. This reduces the risks associated with automation as properly working scenarios and functionality is automated rather than the ones which are being built. The below graph (Fig 8. Medium complexity UI Manual vs UI Automated test case) represents how test automation implementation reduces the testing effort in a project.

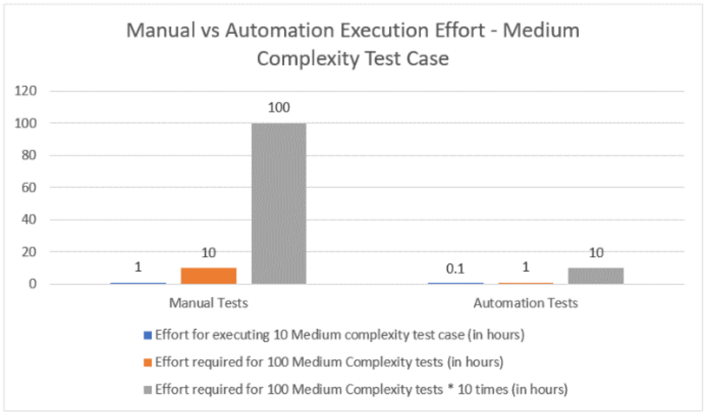


Fig 8. Medium complexity UI Manual vs UI Automated test case

In the above graph a medium complexity UI test case is considered. Medium complexity test case here is defined as test case which contains > 5 steps but < 10 steps. Typically validating one such test case would require 5-6 mins from a manual standpoint. However, if it is triggered via automation, it would only require 20-30 seconds to complete. This applies to one test case. So, when the tests are run manually, in an hour, 10 such test cases could be successfully run. Whereas if it is done via automation that would require only 6 mins to complete. When the number of runs increases, especially when multiple environments are present, this effort reduction becomes very much visible. Hence in the third set the 100 tests are run 10 times which requires 100 hours to complete manually. But in case of automation, it requires only 10 hours to complete. Considering the net reduction in effort the calculation points to an effort reduction of 90% via automation.
When smoke tests and regression tests are automated, it considerably reduces the overall testing effort. However, when such tests are triggered through CI CD pipelines there is an associated cost for maintaining the hardware required to run automation. This needs to be factored in when considering the overall returns on investment.

CI/CD Potential

Continuous Integration happens when small changes are frequently added to a main branch in a version control system. A VCS is basically used to store code in either a central or a distributed environment. Code is added onto a VCS by developers using the branching concept. For each of the feature being developed a corresponding branch would be created which is then combined onto a main branch. From this main branch the product release is carried out.
Continuous Delivery is the process when teams produce software in short cycles with high speed and frequency so that reliable software can be released at any time, and with a simple and repeatable deployment process when deciding to deploy. The CD part is driven by CI process. Hence this goes hand in hand. Continuous Deployment happens when the new software releases occur without any human intervention, i.e., the build to the deployment entire process happens automatically. Some of the major companies like Google, Apple, Facebook, Amazon, LinkedIn employ very mature CI CD process. That is the reason for 99.9% uptime for their software products.
From a test automation standpoint, the CI/CD part is of crucial importance as this helps to run the testing anytime anywhere with complete accuracy. Primarily the code for automation should be stored in a VCS system such as Git, Azure Devops Git, Bitbucket etc. Then this CI component is integrated with a CD tool preferably Jenkins, Azure Pipelines etc which helps to trigger and run the automated tests on a VM machine or devices over cloud and obtain either a success or failure result. Once the tests are successful the development code gets promoted from one environment to another. This helps to ensure that the product remains reliable and at the same time helps to avoid time delays for releases. If the test stage passes successfully the code is auto promoted to the higher environments.
However, there is a cost associated with CI CD implementation as provisioning a full time running VM, network security groups, devices over cloud all these comes at a price. Hence when calculating ROI on automation these costs also need to be factored in to understand the overall profitability of the test automation implementation.

IX. Conclusion

A high-level discussion about test automation is accomplished through this paper. The advantages, the features and the types of automation are covered here. Test automation has a huge potential in a Project lifecycle when implemented based on the discussion above. It would result in huge time savings whereby giving adequate savings on the budget as well. The potential for implementation of CI CD and how it affects the overall automation cost is also covered at a high level. The automation triangle is an effective approach and based on that a strategy could be developed during the project instantiation phase. When that strategy is effectively implemented the team can reap in the benefits of the same.

Acknowledgments

Thanks to all the people who had given me opportunities to work with different projects and helped to understand different types of automation practices followed. Also, I would like to thank Neudesic, an IBM Company for backing me with resources and experience which helped in completing this work. Finally, thanks to the team who had developed the format of this paper.

REFERENCES

[1] Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, Automated Software Testing - A Study of State of Practice, School of Computing, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden

[2] Milad Hanna, Amal Elsayed Aboutabl, Mostafa-Sami M. Mostafa, Automated Software Testing Framework for Web Applications, International Journal of Applied Engineering Research

[3] Sikender Mohsienuddin Mohammad, Automation Testing in Information Technology, International Journal Of Creative Research Thoughts

[4] Raghukiran Basavaradhya & Praveen Narayan, Test Automation Trends in 2020, Trigent Software Inc.

[5] B. A. Kitchenham, S. Charters, ―Guidelines for performing Systematic Literature Reviews in Software Engineering,‖ EBSE Technical Report, Keele and Durham University, 2007.

[6] T. Kanstrén, "A Review of Domain-Specific Modelling, Software Testing," The Eighth International Multi Conference on Computing in the Global Information Technology, 2013.

[7] V. Sangave and V. Nandedkar, "A Review on Automating Test Automation," International Journal of Advance Research in Computer Science and Management Studies, vol. 2, no. 12, 2014.

[8] V. Sangave and V. Nandedkar, "Generic Test Automation," International Journal of Science and Research (IJSR), vol. 4, no. 7, 2015.

[9] M. Monier and M. M. El-mahdy, "Evaluation of Automated Web Testing Tools," International Journal of Computer Applications Technology and Research

[10] "Test Automation for Web Applications," 2017. [Online]. Available: http://www.seleniumhq.org/.

[11] A. Ema and E. M. Reddyb, "Software Test Automation: An algorithm for solving system management automation problems," International Conference on Information, Communication Technologies (ICICT), vol. 46, pp. 949-956, 2015.

[12] Wikipedia, the free encyclopedia, https://en.wikipedia.org/

[13] Mobile OS Market Share Worldwide, https://gs.statcounter.com/os-market-share/mobile/worldwide

[14] Your automation testing strategy: pyramids, triangles and beyond, https://www.mabl.com/blog/your-automation-testing-strategy.

[15] T. Kosa, M. Mernikb and T. Kosarb, "Test automation of a measurement system using a domain-specific modelling language," Journal of Systems and Software, vol. 111, p. 74–88, 2016.

AUTHORS

First Author – Muhammed Suhail TS, B.tech, Computer Science and Engineering working as Senior Automation Consultant, Neudesic, an IBM Company, suhailts@gmail.com, +91-7736632677

Correspondence Author – Muhammed Suhail TS, B.tech, Computer Science and Engineering working as Senior Automation Consultant, Neudesic, an IBM Company, suhailts@gmail.com, +91-7736632677