INDIUM

Behind the scenes

- A script evaluation walkthrough

Prepared by: Kavitha P.R.

Table of Contents

1 Abstract

Are we in the era, where a "Pass" is just taken by face value? Unfortunately not. We are to provide huge capacities of test evidences, proofs, witnesses and what not to be sure of things. Be it a man-made test case or a machine generated automated script a "Pass" has to be rightly validated.

When we are on the testing arena, as we all know automation has taken up a lot of space. For a successful automation project we cannot stress enough that the automation engineer and the manual tester should work together. Usually an automation engineer's attention is on an optimized script - focusing on how fast it runs, how effectively the loops are handled, how it fetches the required data, etc. While the automation engineer's role handles these important aspects of an automation script, it becomes the manual tester's responsibility to ensure coverage and quality.

Several processes can be followed to help evaluate a script successfully and in a streamlined manner. There are checklists that can be maintained, metrics that can be tracked, data sheets that can be collected and most importantly good readable / easy to understand test reports are to be generated which is meaningful to the layman. All that said, there may arise several gaps in what is expected out of a manual tester. There can be instances where one will also have to take care of handling technical disabilities in understanding the code / database and the other such units behind the scenes.

Key Takeaway:
The process of manually evaluating an automated script is not a new phase but are we as testers ensuring completeness of this script? This is the motive behind this paper.

This paper aims at:

- ✓ Defining a manual tester's role in tool evaluation
- ✓ Identifying what each script reads and its type
- ✓ Uncovering hurdles and challenges with possible solutions on how to achieve complete coverage and quality
- ✓ Helping create checklists such as - on when and where to begin this process and others

2 Automation in the arena of testing

Process, technique, best practices, what not? Eventually it all boils down to satisfying the customer. In today's short application development life cycles, is there a smart way to test all the test scenarios manually? Not at all. If an application needs to run on multiple platforms, then the cost and manual effort required for testing is more tedious and time consuming. In order to step up the testing cycle and to increase the test coverage and efficiency of a test, automation testing is required. Test automation allows performing different types of testing efficiently and effectively on various platforms and various configurations. When we take a step back to think about data, we will agree this is tricky. We can pass multiple sets of input across various platforms by using various approaches such as parameterization, grids, etc. which helps achieve more regression in a short span - thereby improving the scope of testing greatly.

3 Why validate an automation script?

Software mishaps are not new. From United Airlines handing out free tickets to Dropbox's outage, we as testers have heard several incidents on how software testing / validation is a must.

We all know the importance of automating the manual tests. This certainly saves the cost, manual testing effort and speeds up the delivery. But how many of us know the importance of validating the automated scripts? A 'Pass' result on script execution does not determine the success of the automation. Test evidences are what that speak. The success of the automation is to ensure all the verification points defined in the manual tests are captured by the script. Validating a script is proof (or evidence as we technically call it) that a test case has been rightly automated.

3.1 Importance of validating an automation script and the role of a manual tester

Having that short but powerful discussion on the need for automation, it is important to discuss how we ensure the automation effort / deliverables are a success. This is where we start focusing on script validation. This phase plays a vital role in the automation process. The main objectives of automation script validation are:

- to ensure all the validation points are covered
- the right check points are used
- to capture all the relevant inputs needed to optimize software quality and ensure that the application meets business requirements
- to make sure that the scripts are scalable. This can only be achieved if one ensures that the scripts are written with clarity and are easy to maintain.

As a manual tester understands the application behaviour the best and has sound knowledge in the domain, he / she is the right choice to perform the Automation script validation. Manual validation has a significant place in adoption of software techniques. Certain elements must be manually validated - that's never going to change. There is a tremendous benefit to this process, and automated validation does not seek to replace it.

3.2 Effects of not validating the automation scripts

We have heard several mishaps when testing is not performed right. It is important any application is tested completely before is it applied to the public. This becomes more important when the application is related to life and death like in healthcare / life sciences domains. Banking is no exception either. Every domain is important by its own. That being said about testing, when we start relying on an automated system to do it for us, the pressure becomes even more. Yes, automation saves time and cost. But this is only when the automated system is well validated and is in place. When a system is tested manually we can expect the tester to think wise of scenarios outside the happy path but an automated test is not the same. All the thinking should go into it before it is completely trusted on. This again is in the hands of a manual tester.

4 The script - Behind the scenes

4.1 Nature of a script

A good automation script should follow some common characteristics such as:

- Coverage: All the validation points should be covered in the automated test scripts

- Execution time: Wait command should be used only in the required area. Unnecessary waiting time may lead to consume more execution time.
- Planned: The automated script should follow the quality based on the size, coding standards, maintainability and level of re-usability. The organization should possess the pre-defined standard structure to ensure the quality. It should comply with applicable industry standards and regulations.
- Short and crisp: Having more test cases are not always good. It takes more time to develop, execute and maintain them. Optimization is more important to ensure enough tests without too many things to manage.
- Standards: Proper naming conventions will allow users to easily understand the source code. The potential benefit of naming convention is to understand the code better, so that the code can be reused after a long interval of time. The name of the scripts should be consistent and common. It should not have any specific terms related to the organization.
- Understandable: Proper commenting should be placed wherever required to avoid confusion. Comment may include author name, script created date, version number, last modified date, last modified user name etc. and avoid extra coding which is not in use. This extra coding may lead to confuse.

4.2 Advantages of a well-structured script / validation

There is no stressing enough that a structured automation approach which includes both scripting and validation is required to achieve delivery of high quality. This will guarantee:

- Cost efficiency
- Easy maintainability
- Re-usability
- Flexibility
- Reliability
- Robustness
- Portability

In the upcoming sections, we will touch base to see how these parameters have improved with our script validation solutions.

5 Hurdles, challenges and solutions

Though the term script validation might sound simple it may not be as simple as it sounds. There are certainly a few hurdles as in any case.

5.1 Hurdle and Solution 1 - Technical disability

Technical disability may arise in several forms:

- Scripting knowledge

Scripting is an art. The more you are in touch with the better you get at it and your creativity grows. A manual tester will understand the logic, approach, looping, etc. but may not be as tech savvy many a times to understand every aspect of every script such as the syntax, advance concepts in coding, etc.. This becomes difficult when thorough script validation is needed.

- Tool knowledge

In many cases with all the tools being introduced in the market every day, even for an automation engineer, it is difficult to keep up with it. But he can be trusted with his technical expertise to get acquainted with the tool quickly. This arises as a challenge for a manual tester as his nature of work is usually different. But we know for effective script evaluation, the tester should possess the knowledge of the tool used for automation. The tester should be able to understand the process on how the tool has been handled, the framework used, etc..

Solution:

Before engaging into the script validation, training or a KT session should be planned and conducted systematically between the manual and the automation testers. This will help the manual testers to attain knowledge about scripting and tool handling. Considerable time should also be spent on helping the automation engineer understand the application under test. The hours spent here may seem like additional cost but better to fix it here rather than try fixing this at a later phase. As we know cost gets higher as we get closer to a release.

5.2 Hurdle and Solution 2 - Test data availability

Determining appropriate test data is a core part of test environment set up process. It's a manual tester's responsibility to pick the relevant data with respect to the defined scenario. It is not possible to create an entire set of new data on every execution cycle. Therefore the tester should choose wisely the best data that is apt for a scenario - ideal in terms of cost and time.

Solution:

Data validation is the process of ensuring that an application functions on clean, correct and useful data. Ensure data sufficiency. A suggestion would be, even before the script comes for validation, as it is being created, plan time with the automation engineer to decide on a test bed. Think of good data like using the ones from production, various live databases, etc. Also decide how you would like this data picked. This is where you can chose between parameterization or databases fetches. Do not fail to remember that the data set can tend to get huge therefore plan to clean up as part of every execution's end.

5.3 Hurdle and Solution 3 - Inconsistent script behaviour

5.3.1 Synchronization

If the scripts are not synchronized, it would certainly affect the test execution as a script depends on many external factors. The obvious solution here is Wait and Sync statements.

Solution:

While validating the script make sure that the script is formed with commands like Wait, Wait until and also having synchronization commands wherever required. This should be dealt with cautiously, taking care that these statements are used only when absolutely necessary. The important thing that needs to be avoided is to not introduce unnecessary wait times or arbitrary sleep variables to synchronize test scripts with the application as this may slow down the execution time. This comes easy with a manual tester's experience with the application - the loading time of various scenarios, etc. are to be thought of. Verify using the break points and ensure wait/ synchronization commands are used perfectly in the scripts.

5.3.2 Environmental factors

Environmental factors may not be in our control all the time. Here are a few common ones:

- Internet speed
- Independent execution PC
- Browser security settings
- DB connection
- Browser cache issues
- Running the scripts in an unsupported environment

Solution:

Ensure the below points are covered in scripting to overcome the environmental factors.

- Automation scripts may not be able handle certain pre-requisite such as application / environment setup. Ensure necessary checklists are maintained and these are part of the Execution guides / reports.
- Diligently check if all queries are optimized.
- Understand how indexed tables are used in the application under test (if any) and think of ways to handle this best in scripts.
- Suggest DB normalization

5.4 Hurdle and Solution 4 – Hard coded data within scripts

Hard coding is not a good practice. The hard coded data in the source code may get cumbersome to maintain. If the value is hard coded in multiple places within the code, the developer should remember all the places the hard-coding exists and whenever any change is made, the hard-coded data needs to be changed in all the places. Most importantly, if the value is hard coded the test cannot be performed with a different sets of input values.

Solution:

Hard-coding or fixed formatting of the data doesn't work anymore. Ensure that the data are called from the external source/ soft code. This can be verified across various break points in the executed automated scripts or results log files. The trick here is to understand the application so well that one knows which data is hard to fetch. There are lots of chances these get hard coded in the script. Spot check scripts on such scenarios.

6 Checklists and other best practices

This section deals with the checklists and best practices that can be followed for the successful script validation. These when followed religiously will also turn out to be good test evidences.

6.1 Checklists

6.1.1 When and where to begin formal validation of the scripts - Entry and Exit criteria

One should begin the formal automated script validation process with a base lined version of scripting for confident delivery and should ensure there is a formal checklist for Exit and have an internal sign-off.

6.1.2 Group your scripts

Create clever batches to help successful / faster execution of scripts. Batches based on the following can be created:

- Order modules based on data set-up dependency
- Smoke / Sanity suites
- Area-wise suites

6.1.3 Traceability

Just like for manual test cases, even for automation scripts, unbreakable traceability is one of the most essential items in the check list. There should be a proper handling in traceability between the requirements and automated scripts as well as between the manual test scripts and automated scripts. This can be a simple spread sheet as below or there are several online systems that will help keep track of these details (A few common test management tools that are popular are QC, TFS, Test Manager, etc.). As part of the traceability, also maintain status, screen shots, etc. which will help when you are preparing your test summary reports or are finally computing your metrics as proof for testing.

S.No	Manual Test case ID	Script ID	Script Version	Prerequisite	Missing Coverage	Validation Status (<in build>)	Screen Shot for Failed Scritps	Comments
1	CPN-52.MyA01	TestCase001_RegrMyA01	v0.11	NA	-Color validation	Passed (6.00.8)		-Step 4 - Unable to capture the text "Record successfully saved" because it flashes too quickly
2	CPN-52.MyA02	TestCase002_RegrMyA02	v0.12	NA	None	Failed (6.00.8)	TestCase002_RegrMyA02 - Step 6 - Failed.jpg	
3	CPN-52.MyA03	TestCase003_RegrMyA03	v0.5	NA	None	Passed (6.00.8)		
4	CPN-52.MyA04	TestCase004_RegrMyA04	v0.14	-Step 7 - Requires an open vendor return	None	Passed (6.00.8)		
5	CPN-52.MyA05	TestCase005_RegrMyA05	v0.6	NA	None	Passed (6.00.8)		
6	CPN-52.MyA06	TestCase006_RegrMyA06	v0.16	NA	-Values in PDF	Passed (6.00.8)		-Step 3 and 4 - Values in reports are not validated
7	CPN-52.MyA07	TestCase007_RegrMyA07	v0.17	NA	None	Passed (6.00.8)		
8	CPN-52.MyA08	TestCase008_RegrMyA08	v0.4	NA	None	Passed (6.00.8)		
9	CPN-52.MyA09	TestCase009_RegrMyA09	v0.1	NA	None	Passed (6.00.8)		
10	CPN-52.MyA10	TestCase010_RegrMyA10	v0.2	All open Excel sheets should be closed before running this script.	None	Passed (6.00.8)		

Figure (i) - Traceability Matrix

6.2 Process – Version control

Maintaining versions is the biggest challenge in automation script validation. Using version control tools we can avoid confusion across different versions. This will help track the changes made to test scripts and one can be sure that the latest has been validated.

6.3 Reports

Customers though kept informed at every phase of development, it is finally the test report he or she will be interested in. This is the green signal or indication that the product is good to go. His interest lies there and more than on anything on a defect free product. What proves a product is defect free? It is eventually the reports. He will not be getting into the code or the script or the test case. Though quality has to be ensured at all these stages, the final summary of testing proves it all.

Tracking is very essential at all phases of the project. How do we make sure this? The answer is – communicating formally – status reporting! Test report is a communication to establish transparency between the QA team and the others which includes both test cases run and their status.

Reports should basically say it all. Take the liberty to define your report. Finally it is only the reports that everyone is going to look at. Some useful pointers maybe:

- Consistent messages
- Pay attention to grammar and spelling
- Appropriate screen shots with highlights if applicable (Some domains require test evidence even for Passed scripts!)
- Alerts and pop-ups are well captured – there can be difficulty in capturing values in a bit mapped alert. The best way to capture value within would be directly from the DB.
- Every check point is marked with an easy colour or tick/cross

One step higher linking results to the respective cases. This way if something fails, it is easy to go to point that requires debugging.

7 Case study and supporting facts

Here is a case study to support the solutions explained in this paper and how it guaranteed in higher quality in better times. A similar set of scenarios were automated in two different phases. Lessons learnt from Phase I were the key behind identifying these best practices to be implemented for Phase II.

<table>
<tr><td colspan="2">Domain: Retail

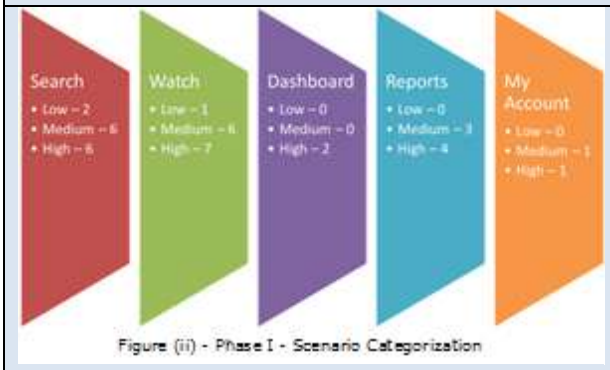
Scope: Automate identified set of scripts</td></tr>
<tr><td>Phase I</td><td>Phase II</td></tr>
<tr><td>Total Scripts: 39</td><td>Total Scripts: 42</td></tr>
<tr><td> 

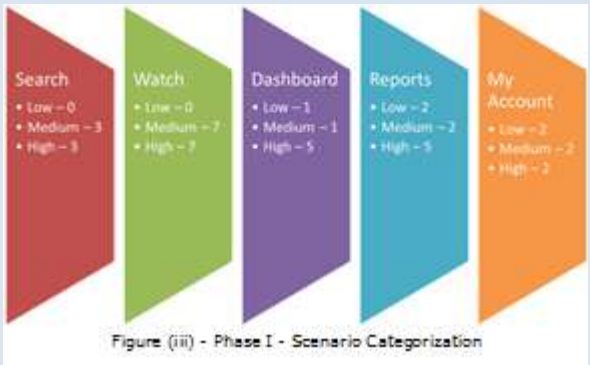
Figure (ii) - Phase I - Scenario Categorization</td><td> 

Figure (iii) - Phase I - Scenario Categorization</td></tr>
<tr><td> 

Figure (iv) - Phase I - Complexity in %</td><td> 

Figure (v) - Phase II - Complexity in %</td></tr>
</table>

Automation effort was carried out for this project in two phases. The phases were quite similar in effort considering the number of test cases to be automated in each phase and the complexity of these. Following are a few metrics to support the success of Phase II.

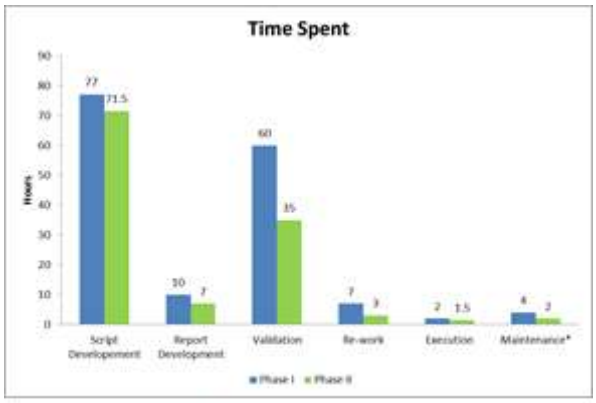


Figure (vi) - Time Spent

Time Spent:

This chart shows the hours spent on all the activities involved in a typical script evaluation project - Phase I versus Phase II.

Significant time is saved on the Execution / Maintenance effort as proper planning / grouping of test cases resulted in a well formulated suite structure.

Schedule Variance:

The building of the Phase II suite reflected right from the project planning.

Most activities were completed ahead or on time because of the structured approach and elimination of ambiguity.

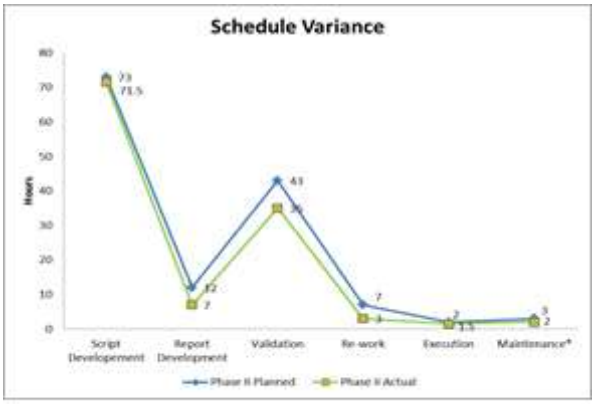


Figure (vii) - Schedule Variance

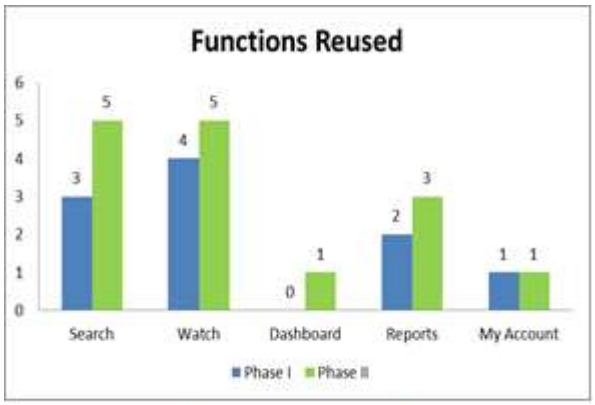


Figure (viii) - Functions Reused

Reusability:

Discussion with the manual tester prior to script development helped plan a function's approach better thereby enhancing the possibility of reuse.

Issues Found in Validation:

Cost of fixing issues in scripts later in the lifecycle of the project is always higher. With initial KT sessions and the understanding / expectation set clear, even before the script hits validation most issues are sorted out or are planned for.

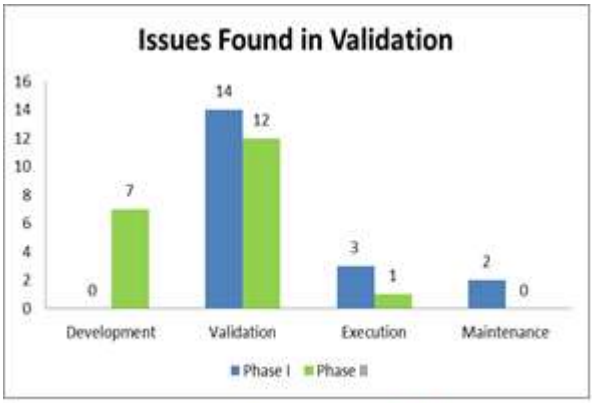


Figure (ix) - Issues Found in Validation

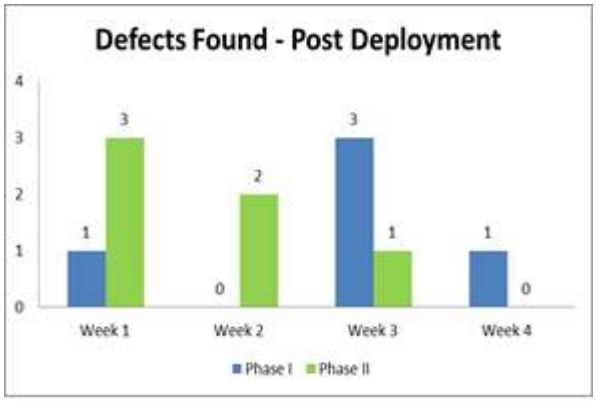


Figure (x) - Defects Found - Post Deployment

Script Efficiency:

The efficiency of a script is found by the number of defects it captures post deployment. (Note: Post Deployment here means the script deployment and not the application deployment to production.)

Deeper scrutiny of scripts performed here resulted in better and more efficient scripts.

Here are the value additions summarized based on the various activities involved in an automation project:

Stages	Phase I	Phase II	Variation	Analysis
Reusability	10 functions reused	15 functions reused	25%	- Well thought of scripts resulting in increasing the scope of reusability.
Designing a Report	10 hours	7 hours	30%	- Much clearer and user-friendly reports as the manual tester defined the reports from an end user perspective. - Saved on de-bugging effort.
Validation	60 hours	35 hours	40%	- Script validation starts even before the formal phase. This helped save time.
Re-work	7 hours	3 hours	60%	- Earlier intervention of the manual tester in the process helped plan

				scripts / data better. - Saved considerable time on re-work effort.
Execution	2 hours	1.5 hours	25%	- Appropriate checks on Wait and Sync statements, optimized queries and normalized databases helped save execution time. - More confident execution on various platforms.
Maintenance	4 hours per week	2 hours per week	50%	- Well-structured / commented and standards followed in scripts resulted in easy maintenance.
Script Efficiency	NA	NA	20%	- Defects found by the script during the initial weeks of deployment proved its efficiency.

8 Conclusion

Automation script validation, when carried out in a planned and effective manner, can offer great benefits and certainly help gain confidence in the quality of the scripts created. Properly planned and implemented automated testing can significantly lower project risk and cost. It can not only reduce the risk of the existing project, but in fact reduce the risk and cost of upcoming projects too. We all know the importance of automation; this write-up is to help understand the importance of script validation. Everyone wishes, in all aspects, to achieve "Best Results". Understand what is happening behind the scenes and this will definitely help you get there.

9 References

- http://www.automationframework.info/
- http://msdn.microsoft.com/en-us/library/ee658094.aspx

10 Author biography

Kavitha, Senior Test Engineer with Indium Software has 5+ years of experience in testing domains such as Banking, Lifesciences and others. She believes in process and converting them to best practices. This has helped her write this paper on how best one can evaluate the scripts generated as part of an automation effort. She is a confident and cheerful person who loves spending her time reading books and playing with little kids.