

A Model to Quantify and Improve Software Test Automation

Kiran Jammalamadaka¹ and Ramakrishna V¹

ABSTRACT

With test automation being a corner stone for improving quality of releases and with increased reliance on automation, automation framework and scripts being developed needs structured model to measure effectiveness of automation in areas of maintainability, reliability and extendibility. Also a need of a maturity model to guide and improve automation being developed arises. In this paper we proposed a model called Automation Reliance Model which fills in the space of quantitative measurement of automation development, and guides to improve same. This model is derived collating our experience with automation frameworks, script development and ideas from Capability Maturity Model integration ^[5], Failure Mode Effect Analysis ^[6] and Test Maturity Model integration ^[2] and also taking into account standard quality measures followed for Agile based implementations.

Automation success depends not only on tools used and test harness, but on entire life cycle of implementation. Automation Reliance Model focuses on 3 phases' requirements, implementation (Design and development quality) and maintainability of automation. Automation Reliance Model defines a scale for automation quality though quantitative measurement based on standard metrics, and provides direction to improve.

Keywords: Test automation, measurement, maturity model, quantitative model, improvement model.

INTRODUCTION

Organizations are relying on automation for increasing test coverage, ability for faster feedback cycles thus improving overall product quality. However, quality of automation being developed plays a major role in achieving these objectives. Organizations traditionally evaluate the automation framework's quality by no of faults detected and test coverage achieved. Often it gets too late if any fault has been uncovered in the framework itself; any constraints resulting in poor maintainability of the automation; hence evaluating the framework ahead has more significance.

There is a definite need in identifying a quantitative method to measure and improve the overall automation experience. Few models have been proposed like Test Process Improvement ^[1], TMMi ^[2] etc. These models however are lacking enough detail for test automation ^[3].

Maturity Model

Organization started depending up on the computerized information systems in the early years of 1960, and significance of software development grew ^[7] accordingly, as the software development is in early stage failure was a common pain for most of the organizations and delivering the project with in the budget, on time with quality was a major pain area. Many researchers and organizations were established and funded to address the problem, one such organization funded by United States Air force is Software Engineering Institute (SEI) ^[8].

The need for model is evident to address the below but not limited to the following

¹ K L University, Andhra Pradesh, India, E-mails: jvskkiran@gmail.com; vramakrishna@kluniversity.in

Communication: Model improves the communication or establishes a common language between the parties in order to understand and resolve the disagreements.

Gives bigger picture: Model helps to understand the organizations' current process and also let everybody know the directions

For a couple of decades, industries and government organizations struggled to deliver, however very few organizations could deliver with in the budget and time line, after careful study of the success organizations process, it was observed that the successful delivery of the project is due to some heroic efforts of the team members and real struggle is repeating and predicting the outcome.

- 1) CMM Model: The capability maturity model is a methodology used to improve the organizations' existing process in order to improve the product quality. Watts Humphrey's published his Capability Maturity Model (CMM) in 1988^[9]. CMM was used initially as an evaluation tool and for software government contracts and later adapted to other organizations.

As M. Imai, about Kaizen mentioned that Improvement cannot be achieved in a major step yet to be achieved with small incremental steps ^[10], CMM provides a framework to organize these steps in order to achieve the process improvement.

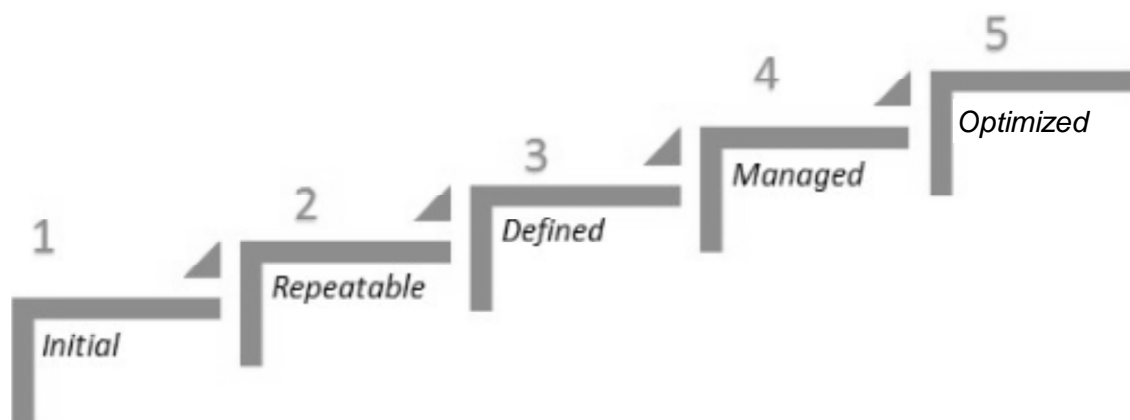


Figure 1: CMM Maturity Model.

CMM proposed a 5 maturity levels to evaluate and calculate the process capability, a maturity level is a step towards achieving the process maturity.

Initial: Process as an ad hoc and success is due to a few team members.

Repeatable: Success is repeatable and discipline is in place.

Defined: All the activities are defined and standardized in the form of documents and all projects follow an organizations standard process.

Managed: Measures the software process and product, both are controlled

Optimized: Enables the continuous improvements with quantitative feedback from the past and implementing innovative ideas to solve problems

CMMi is the successor of CMM, and CMMI model Refined and Published as SW-CMM v1.0 ^[4]

- 2) Test Maturity Model Integration: CMM and CMMi are more of a guide lines to improve the development process and give the direction towards improvement and also determines the maturity of the organization. However, CMM and CMMi are not helpful for the test community since the focus on the testing from these two are less and very minimal, since around 40% of budget and

efforts being spent on the testing more focus is essential, a very high level mentioning about the testing is not sufficient.

To bring more emphasis on the testing maturity, TMMi foundation has created its own maturity model. Test Maturity Model focuses on the test process improvement and compliments CMMi.

Like CMMi staged representation, Test Maturity Model also uses the maturity level concept for process evaluation. TMMi helps to improve the test process, hence impact the product quality in a positive way.

The process areas and level have been represented in the figure 2 below,

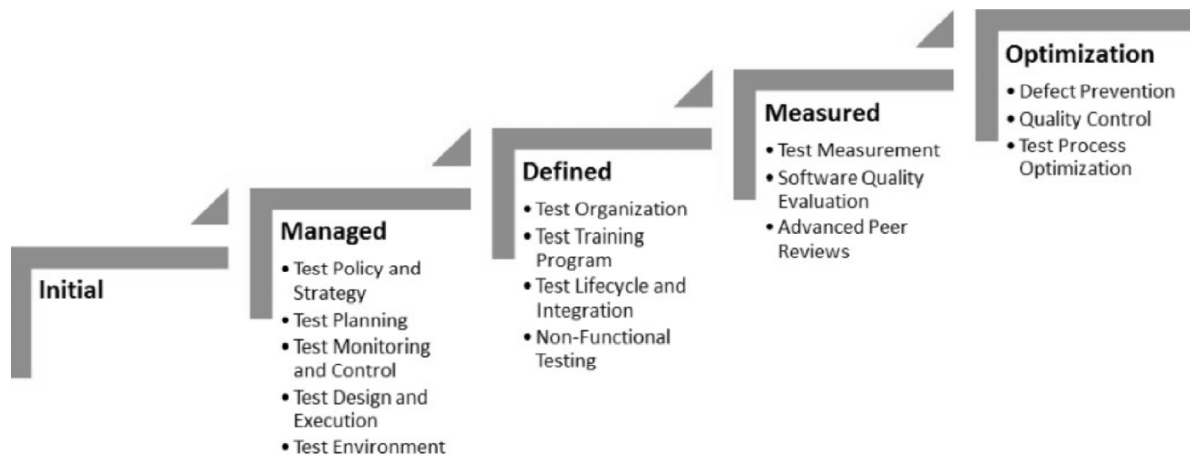


Figure 2: Test Maturity Model levels and process areas

TMMi considers tools as a supporting process, for example an organization is using a test case design tool to generate test cases, TMMi does not consider use of tools even for non -functional at maturity level 3.

To summarize, TMMi provides a few standards to the process and also provides the direction to improve, by implementing TMMi organizations can know where are their current test maturity and what should be done to reach next levels.

Summary of the Models

Table 1
Summary of Maturity Models

Maturity Model	Notes
CMM	Focus on the overall development process, no mention of testing practises
CMMi	Mentions about testing as Verification and Validation, not much discussion about testing other VER and VAL
TMMi	Focus only on the testing, tool have been considered as supporting processes, and not mentioned how to do.
MPT.BR ^[11]	Mentions about the test process and test tool mostly automation, however it is not driven by the generic metrics.

LITERATURE REVIEW AND MOTIVATION

From the above discussion, it is evident that focus on the testing and automation are also vital. This led many researchers to focus on the testing and test automation and proposed a few models.

Gelperin and Hayashi^[12] in TMM described that the primary issue with test ware is reuse, in order to support the reuse an eco-system needs to be established with configuration management and modularization and independence, they also recommend to automate the test environment.

Jacob^[13] made emphasis on TMM and reiterated the significance of testing, however not recommended and new requirements to the TMM.

Adaptions were made to TMMi^[14] to address the new needs of agile, however test automation advancement not proposed.

Test Automation Improvement Model^[3], is closely related to the work we are doing, TAIM also focuses on the quantitative measurement of the automation by specifying a few generic areas [GA] and Key areas [KA].

The overall result will be derived by combining and aggregating generic areas and key areas. TAIM quantifies the maturity at very high level.

AUTOMATION RELIANCE MODEL

Taking influence from CMMi Process Area categories for different types of activities, 3 focus groups were identified towards automation maturity. Requirement Management, Requirement Development, Technical Solution process areas are adapted/customized for automation development and maintenance activities.

CMMI's Process Area concept is further expanded to support Automation Reliance Model in measurement of automation maturity.

Automation maturity levels are classified based on stability, maintainability and effectiveness of automation.

First level: Being quick and dirty automation, where focus is on short term benefit of getting automation developed and executed.

Second level: Focuses on automation being repeatable and extendable. Basically, teams think in terms of design to implement and try to adapt frameworks or design patters up to some extent, may not be fully implemented to get the automation solutions.

Third level: Focuses on consistency of the automation, and starts looking at effectiveness of automation maintainability. Automation is considered in with code development and as in line with Code development, test automation is expected to have practices of error handling, debugging which will help reduce maintenance.

Fourth level: Focuses not just on the development of automation, but also with respect to how well the automation execution is also prioritized. Focus is on long term strategy of automation development, starting from right automation tests being identified and efforts to balance the tests and reduce the duplication of tests (UI Tests, Middle layer tests or API tests and Unit level tests), test sets are classified appropriately based on the change sets and leveraged through continuous integration practices. High level of automation reliability is also expected in this level.

Fifth level: Can be considered as auto pilot of automation being developed. Automation is highly reliable, with minimum maintenance and precises error identification using design approaches to isolate suspicious code changes^[15] like techniques. This can be integrated to Continuous integration into Continuous Deployment as well.

ARM also helps in providing guidelines for automation planning which is critical aspect of leveraging automation. Teams can focus on key practices to be followed to ensure automation success.

Key Focus on ARM is to understand the key impact areas for effective automation development and usage. A quantitative mathematical model is derived to help teams understand and improve the automation development continually. Mathematical model focuses on having normalization across impact areas to ensure there is standardization different automation types. And also helps to have a common understanding

of measurement, and also level which will be very helpful when working with multiple teams. The Mathematical model is only to help teams quantitatively measure and improve the automation, rather to analyse the mathematical equations. Mathematical equations and weightages are derived purely from a collated experience of multiple people developing, leveraging automation which is subject on refinement based on future studies.

Key practices (Focus Areas) under the respective process areas (Focus Groups) are further (Focus Areas) detailed. Metrics to measure and improve these focus areas are derived.

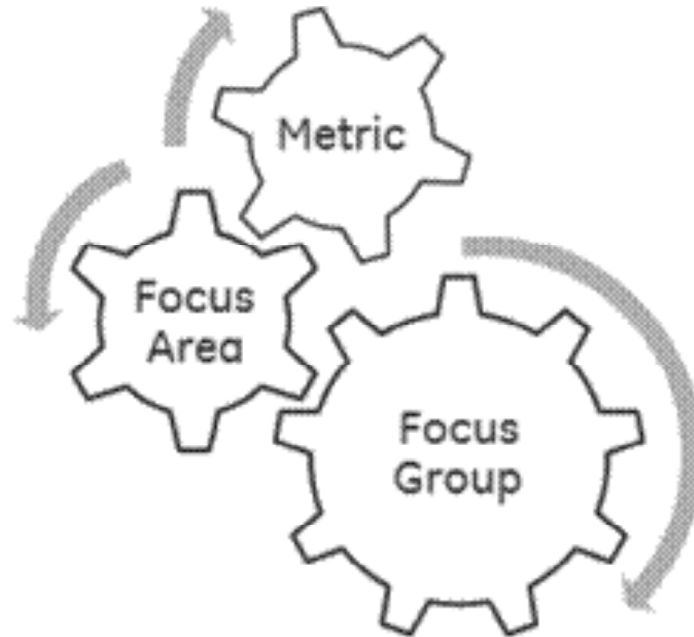


Figure 3: Representation of Focus groups, Focus areas and Metrics

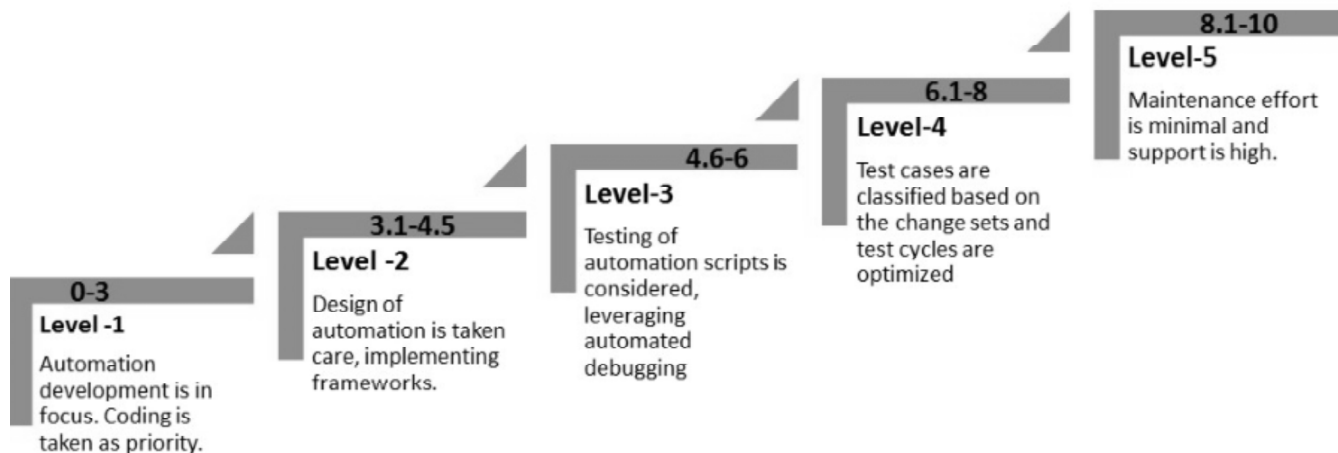


Figure 4: Quantified Levels of ARM

Focus Groups: ARM consolidates phases of automation implementation into 3 focus groups (FG), requirements, development and maintenance of automation.

Taking experiences of SDLC patterns, contribution of Requirements, development and Maintenance towards quality of deliverable is considered to be at 20:60:20. Same is considered as weightage for respective focus groups.

Table 2
Focus groups and weightages

<i>Focus Group</i>	<i>Weightage</i>
Requirements of the scripts	20%
Development of the scripts	60%
Maintenance of the scripts	20%

Each focus group is further broken down into Focus areas to measure the value achieved by respective focus group, called as Process Value Contribution.

Based on the weightage for focus group and considering the process value achieved, ARM classifies automation into various levels.

Mapping of Focus Groups and Focus areas, explained with an example. In the below table

Table 2
Levels of automation with examples

<i>Focus Group</i>	<i>Focus Area</i>	<i>Weightage</i>	<i>Area Value</i>	<i>Weighted Value</i>	<i>Normalized weighted value</i>
Requirements	Test case optimization	10.0	10.0	100.0	0.1
	Optimized test cycles	10.0	100.0	1000.0	1.0
Implementation	Design	15.0	59.9	898.0	0.9
	Coding and Unit testing	30.0	77.1	2312.2	2.3
	System Testing	15.0	93.3	1400.0	1.4
Maintenance	Documentation & support	10.0	100.0	1000.0	1.0
	Maintenance Effort	10.0	12.0	120.0	0.1
Total Weightage		100	452.3	6830.2	6.8

In the above example, Sum of Normalized Weighted Value (6.8 as above) depicts the ranking of automation script based on quantitative mechanism. Automation scripts are ranked as LEVEL 4 for 6.8 weighted values.

Focus Areas

Each Focus group is further translated into Focus areas which are representation of process steps or practices followed towards automation implementation. Each of these focus areas have a quantifiable metric.

Focus area is further broken into practices and contributing value of each of them. Each practice is measured based on a quantifiable metric, through which process contribution value is measured.

Normalization of metric values is done to derive area value for a focus area.

As explained in the above Table 2, Requirements focus group has two focus areas viz.

1. Test Case Optimization
2. Optimized Test cycles.

These are further breaking into the following

Table 3
Focus areas and Priorities

Test case Optimization	Test Cases are organised with proper naming convention	1	33.3
Optimized Test Cycles	Test Plan is optimised on to number of test cycles being run with associated test cases	2	66.7
	Test groups are visible with respect to the feature releases and service packs	1	33.3

Respective metrics are evaluated either manually or via automated methods to determine a process contributing value. Automated methods are suggested to compute the metric value.

Contributing value for Focus area is determined based on weightage of each metrics on the focus area (practice).

Refer Appendix A for details.

Normalization

Automation success is not evenly distributed across Focus Groups and Focus areas. Each Process Step has a unique contribution to overall success, Design might be a higher contribution rather unit testing. Automation Reliance Model normalizes this distribution by adding weighted contribution across Focus Groups and Areas.

Considering Automation success as a measure of 100, Focus Groups are distributed at 20:60:20. Focus areas further divide Focus Group contribution as per influence of respective Focus area. Ex: Requirements being 20, Test case optimization and Optimized test cycle will be 10 each.

Metric Weightage derivation

Let's assume we have "N" metrics in a Focus area, All the N metrics should be prioritized in the order of highest to lowest. i.e., Priority 6 will be more important than priority 5 and so on.

Each focus area is distributed for weightage of 100, across any number of metrics available.

As mentioned above; considering N number of metrics; we will have X as minimum weightage across them as below

$$100 = x \cdot N_1 + x \cdot N_2 + x \cdot N_3 \dots \dots \dots x \cdot N_n$$

$$x (N_1 + N_2 + N_3 \dots \dots \dots N_n) = 100$$

$$x (n (n+1)/2) = 100$$

$$x = (2 * 100) / n^2 + n$$

Considering we have two metrics, we shall have the minimum value as 33.3 ($200/6 \sim 200/4+2$)

Metric with priority 2 will have weightage of 66.7 and metric with priority 1 will have weightage of 33.3

Above derivation gives flexibility in adding a new metric and also adjusting weightages through priority.

Note: Current model does not expect two metrics to have same weightage. In which case, we will have to manually adjust weightage values.

Sum of all the metrics under one focus area needs to be 100, which makes calculations easy.

Evaluation of the model

Based on our experiences and interactions we had with other peers, we have proposed the above focus groups and data points.

Our model ensured a highly transparent and quantifiable model for tracking and improving automation. As part of engagement with vendor teams towards Regression Test Suite development, this model was used to set standards for automation scripts delivered.

Minimum level is agreed and tracked as part of Exit Criteria for deliverables. Automation scripts quality was continuously improved with mapping to focus areas.

With this experience, we believe Automation Reliance model will be quite helpful in measurement of automation quality, and coming up with plans to improve. This also forms as a standard to communicate across teams working on Automation. All teams will be on same page and also can collaborate in achieving higher-level s in automation and to improve overall Automation Experience.

Refer to the Appendix B for detailed set of data points.

CONCLUSION

Automation Reliance model establishes a framework to understand the automation impacts, and acts like a guidance in planning effective automation. Focus groups and Areas provide the success criteria which can be used to quantitatively measure the automation, then to target and improve. This helps as a tool to understand the direction to develop, leverage automation with minimum maintenance and greater power to leverage and benefit the releases. Automation Reliance Model is built on empirical way with collated experience and understanding to practices being followed.

Our experience with this model was good with the experiments being done especially during the communication with vendor and other team which were geographically separated, and helped us in developing and standardizing the automation and improving specific focus areas. Having said the model requires constant inspection and then adapting to the context of the test automation being developed.

ACKNOWLEDGEMENTS

We would like to thank Devicharan Garigipati for his helpful insights.

FUTURE WORK

Study and Refine to make the model more precise, and also to adapt for varying automation approaches and technologies being used, like Web, Customized & Non Customized software's, Mobile and Cloud Platforms.

Set of processes towards Focus area would also have distributed contribution to success of the respective focus areas. Weighted average for a metric is computed, considering the influence of respect process step on the overall focus area.

REFERENCES

- [1] T. Koomen and M. Pol. "Test process improvement: a practical step by-step guide to structured testing." Addison-Wesley Professional, 1999 ISBN: 978-0201596245.
- [2] E. van Veenendal, "Test Maturity Model Integration TMMi Guidelines for Test Process Improvement", UTN, ISBN 978-9490986100 (2012).
- [3] Sigrid Eldh, Kenneth Andersson, Kristian wiklund "Towards a Test Automation Improvement" Model (TAIM)" 2014 IEEE International Conference on Software Testing, Verification and Validation Workshop.
- [4] SEI, "Capability maturity model integration(CMMISM)", version 1.1/Staged representation", 2002.
- [5] https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration
- [6] https://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis

- [7] https://en.wikipedia.org/wiki/Capability_Maturity_Model
- [8] Mark C. Paulk, Bill. Curtis, Mary Beth Chrissis, Charles V. Weber. "Capability Maturity Model for Software". CMU/SEI-93-TR-24 ESC-TR-93-177 February 1993.
- [9] Humphrey, W. S. Managing Technical People. Reading, MA: Addison-Wesley Publishing Company, 1997.
- [10] Imai, M. "Kaizen: The Key To Japan's Competitive Success". New York: McGraw-Hill. 1986.
- [11] Ana Paula C, Marcos Wanderley G "Towards a Maturity Model in Software Testing Automation" ICSEA 2014.
- [12] D. Gelperin and Hayashi "How to support better software testing". In: Application Development Trends May (1996).
- [14] E. van Veenendal, "Test Maturity Model Integration TMMi Guidelines for Test Process Improvement", UTN, ISBN 978-9490986100 (2012).
- [15] Kiran Jammalamadaka, V Rama Krishna "A Design Approach to Isolate Suspicious Code Across the Builds with Test Traces to Analyze Regression Failures". International Journal of Applied Engineering Research Volume 10, Number 2 (2015) pp. 3559-3568, 2015.

Appendix A

Focus group

- Requirements
 - o Test Case Optimization
 - Test cases are Organised with Proper naming convention (Percentage of Number of Requirements (Test Cases)/ Number of Standardized Requirements (Test Cases))
 - o Optimized Test Cycles
 - Test Plan is optimized on to number of cycles being run with associated test cases
 - Test Groups are visible with respect to Service pack , New feature
 - Implementation
 - o Design
 - Flow of the test cases have been identified
 - Reusable components/functions have been identified
 - Design is complete with respect to test cases being identified for automation. Clear traceability matrix available (Percentage of Number of Implemented Requirements (Test Cases)/ Number of Standardized Requirements (Test Cases))
 - Design uses object Oriented concepts
 - Coupling between components should be in between 1-10
 - o Coding
 - Code completeness with respect to design
 - Object oriented concepts implemented
 - Efferent coupling 1-10
 - Cyclometric complexity
 - Average no of parameters low or med is good (low <3, med 3-5, high >5)
 - Code commenting helps to understand the logical flow Minimum Percentage of Code Commenting is expected to be 20%
 - Variables should be typed, For Ex. Dim xxx as String
 - Exception handling — Clean exits for application and system exception with proper throw Minimum percentage of exception handling code is expected to be 20%
 - Unit Test Coverage for the respective functions (100% is desired)
 - Defect Density
 - o System Testing
 - Negative scenarios are tested for error handling
 - Robustness – 90% success rate of test runs

- Automated debugging process is in place
- Maintenance
- o Documentation and Support
 - Detailed documentation is available on the framework and usage patterns
 - Coding standards and best practices are available
 - User guide is available on framework installation and known issues.
- o Maintenance effort
 - Post Release Defects (Should be zero)
 - Maintenance effort (less than 5%)

Appendix B

Data Points: Data points represent core anatomy of automation script and form contributing points for automation quality.

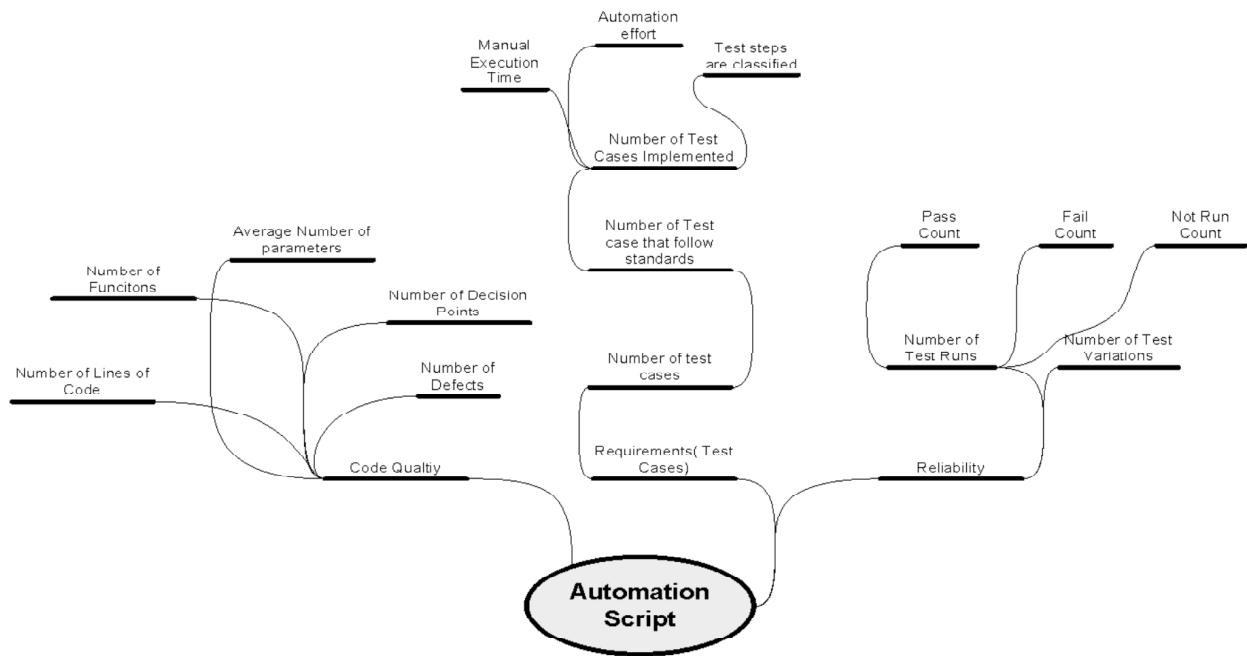


Figure 5: Automation Script anatomy