Using Fuzzy Logic in Manual Test Script Authoring Effort Estimation in Software Test Planning

Ralph Andre C. Roque, Klint Allen A. Mariñas, Michael N. Young, and Yogi Tri Prasetyo
School of Industrial Engineering and Engineering Management
Mapúa University
658 Muralla St., Intramuros, Manila 1002, Philippines
racroque@mymail.mapua.edu.ph, kaamarinas@mapua.edu.ph, mnyoung@mapua.edu.ph, ytprasetyo@mapua.edu.ph

Klint Allen A. Mariñas and Yung-Tsan Jou
Department of Industrial and Systems Engineering
Chung Yuan Christian University
Taoyuan City, Taiwan
klintallen2011@gmail.com, ytjou@cycu.edu.tw

Satria Fadil Persada
Entrepreneurship Department,
BINUS Business School Undergraduate Program, Bina Nusantara University
Jakarta 11480, Indonesia
satria.fadil@binus.ac.id

Abstract

Software testing effort estimation is a critical phase in software project management. Software testing is an integral part of the software development lifecycle (SDLC). It ensures that requirement specifications are met before a product is deployed. Accurate test estimation is crucial to the on-time delivery of a project and ensures that it meets resource constraints. In this study fuzzy logic is used to estimate the manual test script authoring effort in the test process. MATLAB software is used to design the fuzzy logic system that would calculate the duration of the scripting effort. The variables considered were test complexity, test script reusability, and number of test scenarios. The crisp output is the effort estimation in hours. The Delphi method was used to collate the ratings of each expert for the parameters of each membership function in each variable. The linguistic rules were defined after. Crisp inputs for each project were applied to the model and the output was compared with the actual effort duration of previous projects and expert-based estimation. The results showed that the mean absolutive relative error (MARE) of the fuzzy logic model has performed better than the expert-based estimation technique. Thus, the fuzzy logic framework may be used as a decision support tool in manual scripting effort estimation. This study is the first to explore the effort estimation in manual test script authoring. This model could serve as a baseline to test engineers, test leads, project managers, and business analysts.

Keywords

Software testing, Effort Estimation, Software Supply Chain, Manual Testing, Script Authoring

1. Introduction

Software supply chain is any activity involved which affects your software from development until deployment into production (Mcbride, 2021 and Iradier, 2021). In order to produce a high-quality software, it requires an efficient and effective software testing process (Tahvili et al, 2018). Software testing is a time-consuming and costly process, especially manual testing (Tahvili et al, 2018). Thus, effective management is crucial to minimize cost, waste, and time (IBM,2022), to build a competitive infrastructure, and to create customer value (Wisner, 2019).

Software testing is the process of validating that the system meets customer requirements (IBM, 2022). Hence, software testing is an integral part of the software supply chain. It assesses the quality and reduces the risk of failure during operation. The testing process consists of activities such as test planning, analyzing, designing, implementing tests, reporting test progress and results, and evaluating the quality of a test object (ISTQB, 2018). Therefore, a well-defined software testing strategy is required to deliver quality within the planned cost (Shivakumar, 2015). This includes test planning which consists of the estimation of costs, scheduling, and resource's efforts (Dias-Neto & Travassos, 2010). The estimation of testing efforts significantly contributes to the capacity planning of the project which evaluates whether the available resources is capable of meeting the delivery schedule (Wisner, 2019). However, it is still a challenge to define a systematic method of estimating software testing efforts (Bluemke & Malanowska, 2021).

Estimation of software testing efforts has been an area of interest in software development. Abhilasha and Sharma (2013) calculated the number of testers and hours required in regression testing using their Test Effort Estimation in Regression Testing (TEERT) approach. The output of TEERT is in Man-hr which requires the following inputs: Change Type, Number of Selected Test Cases, Test Execution Complexity, and Test Team Productivity. Results showed that TEERT is aligned with the existing test effort estimation method, Requirement Based Test Effort Estimation (RBTEE). In the study of Grover et al (2017), they employed a revised use case point (Re-UCP) model using Fuzzy Technique in software test estimation effort. The Re-UCP formula uses technical complexity factor, environmental complexity, and total actor and use case weights. In manual testing, the research of Aranha and Borba (2009) proposed a method in estimating manual test execution effort based on execution points (EP). The proposed measure, EP, is based on the amount of test actions(steps) and the functional and non-functional characteristics of the application. They have concluded that their method is a viable method of estimating manual test execution. Another study of Tahvili et al (2018) proposed ESPRET (Estimation and Prediction of Execution Time) tool in estimating and predicting execution time of manual test cases based on test specification, requirement specifications, test scripts, and test logs. The ESPRET tool assigns a time value for each test step and sums up the execution time. Empirical results suggest that it can be used as a supportive tool in prioritization, selection, and scheduling of test cases.

Accurate software test estimation facilitates the management of resources and on-time delivery (Nisar, 2018). To improve the use of resources, good testing technique and quality assurance processes are essential (IBM, 2020). This study proposes a process improvement to the manual test script authoring effort estimation being performed during the test planning phase. This activity is carried out whenever there is a minor or major release/modification to the content management platform utilized by a multinational pharmaceutical company. Currently, the test effort estimation technique used is an expert-based technique. Estimation is solely based on individual experience and discretion. By applying fuzzy logic, this study aims to provide a more consistent method of collating the judgements set out by each individual tester.

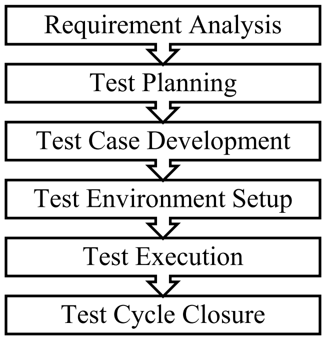


Figure 1. Software Testing Life Cycle

This study aims to utilize fuzzy logic in the estimation of manual test script authoring in the test planning phase of the software testing life cycle (Figure 1). Despite numerous studies in software testing effort estimation, there are little to no studies pertaining to the estimation of manual test script authoring using fuzzy logic. This research is the first to explore the effort estimation of manual test script authoring. This may improve the expert-based test estimation technique utilized by the team and streamline the test planning process by providing a baseline for the estimated duration in authoring a test script. Furthermore, this could provide valuable guidance to test engineers, test leads, project managers, and business analysts in planning the delivery timeline of the overall software project.

1.1 Metrics

Test Complexity pertains to the expected difficulty based on the analysis of the software requirements, modifications, or updates to be tested. It considers the intentions of the user stories and its practical difficulty. More difficult means more amounts of ambiguity and risk factor (Saini & Khatri, 2018). If the complexity is very high, the amount of testing effort should increase as the number of test cases will also increase (Srivastava et al, 2011). This is categorized as easy, average, and complex.

Test Script Reusability is the estimated percentage that a test script from previous projects can be reused to test the upcoming software requirement, modification, or update. This is categorized as minimal, partial, and almost reusable.

Test Scenarios are any functionality that can be tested (QA Madness, 2022 and Guru, 2022). These scenarios are approved by various stakeholders namely business analysts and developers (Guru99, 2022). Testers put themselves in the perspective of the end-user to figure out real-world scenarios and use cases of the application under test (Guru99, 2022). In this study, test scenarios are constructed by the test script author based on the analysis of the software requirements to be tested. Test scenarios is assessed based on the number of expected scenarios. This is categorized as few, average, and many.

Test Scripting Duration is the length of time in hours that the tester spends writing the manual test script. In this study, this is the output of the fuzzy logic model. This is categorized as short, medium, and long.

2. Methods

In this section, we describe the proposed approach for estimating the manual test script authoring effort using fuzzy logic (Figure 2). In this study, MATLAB software application is used to design the fuzzy logic model that calculates the effort estimation. Logic rules and limits of each membership function were provided by the testing team. Three input variables were considered namely test complexity, test script reusability, and test scenarios. The output is the duration of the manual test scripting effort in hours.

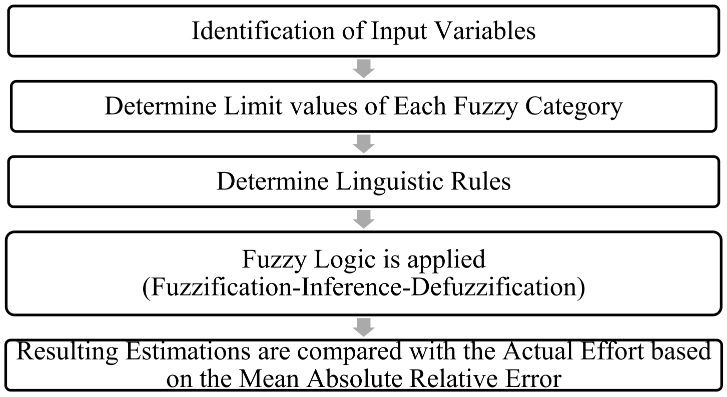

Identification of Input Variables
Determine Limit values of Each Fuzzy Category
Determine Linguistic Rules
Fuzzy Logic is applied (Fuzzification-Inference-Defuzzification)
Resulting Estimations are compared with the Actual Effort based on the Mean Absolute Relative Error

Figure 2. Research Method

2.1 Fuzzy Logic

Fuzzy logic is a mathematical technique used to deal with imprecise, vague, and incomplete data. It is a methodology used for computing with words rather than numbers (MathWorks, 2022). It utilizes the ranges of values provided by humans to emulate human logic (Mittal et al, 2010). Fuzzy logic consists of three stages namely fuzzification, inference, and defuzzification (Malathi & Siridhar, 2011). The fuzzifier transforms crisp inputs into membership values. Inference applies the logic rules in the rule base. Logical operators such as "And", "Or", and "Not" combine fuzzy variables and provides a decision for each rule. Defuzzification combines the output into a single value. Figure 3 shows the general framework of the fuzzy logic system (Taghavifar & Mardani, 2013). The most widely used method of defuzzification is the centroid wherein it takes the center of gravity of the fuzzy set (Kayacan & Khanesar, 2016). In this study the centroid method was used to define the crisp outputs.

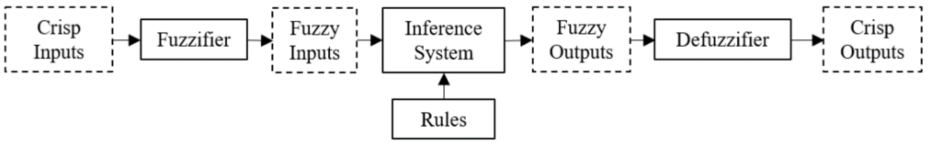


Figure 3. General Framework of Fuzzy Logic System

A fuzzy set is an extension of a classical set. If X is the universe of discourse and x is an element of X, then fuzzy set A is defined on X and written as a collection of ordered pairs. $\mu_A(x)$ is the membership function of x in A. It is written as follows:

$$A = \{x, \mu_A(x) \mid x \in X\}$$

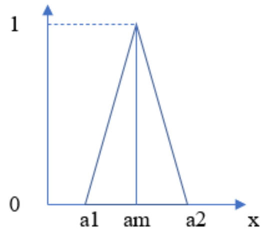
In the fuzzy logic theory, every input belongs to a set of specific membership values (Kayacan & Khanesar, 2016). It allows its members to have varying degrees of membership called membership function having an interval [0,1]. Membership functions can be triangular, trapezoidal, or bell shaped (Mittal, 2010). In this study, triangular and linear fuzzy were used to define the membership functions of each variable. A membership function defines a fuzzy set and can be mathematically written as:

$$\mu_A(x): X \to [0,1]$$

This means membership function $\mu_A(x)$ is associated with fuzzy set A such that the function maps every element of the universe of discourse, X to the interval [0,1].

2.1.1 Triangular Membership Function

Triangular fuzzy numbers are commonly used due to its simplicity. The application is easy and naturally superior (Sarokolaei et al, 2013). Triangular fuzzy numbers are used to identify the limit values for each parameter. It can be represented as $A = (a_1, a_m, a_2)$. a_1 being the smallest possible, a_m being the most promising value and a_2 being the largest possible. (Figure 4).

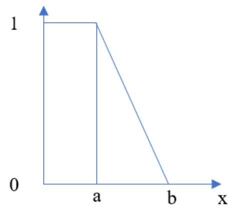


$$A(x) = \begin{cases} 0 \quad if\ x \leq a_1\ or\ \ x \geq a_2 \\ \dfrac{(x - a_1)}{(a_m - a_1)} \quad if\ x \in (a_1, a_m) \\ \dfrac{(a_2 - x)}{(a_2 - a_m)} \quad if\ x \in (a_m, a_2) \end{cases}$$

Figure 4. Triangular Fuzzy

2.1.2 Linear Membership Function

The L-function or Linear membership function is defined by two parameters a and b. If the value of x is less than or equal to a then it is a member of the set. As the value of x approaches b the degree of membership decreases. (Figure 5).



$$A\ (x) = \begin{cases} 1 \quad if\ x \leq a \\ \dfrac{(b - x)}{(b - a)} \quad if a < x\ \leq b \\ 0 \quad if\ x > \ b \end{cases}$$

Figure 5. L-Function

2.2 Delphi Method

The Delphi method is used to form more accurate calculations. In this study, experts were asked to give their ratings based on the smallest, largest, and most promising value for each membership function in each variable. A survey questionnaire was distributed to collect each expert's suggested empirical membership functions for input variables or metrics namely test complexity, test script reusability, number of test scenarios, and test scripting duration. After collating the data, the weighted average was computed. Resulting values of each membership function were then presented to the experts for review. The process is repeated until values were accepted (usually performed two or three times) (Sarokolaei et al, 2013). For every round of estimation, the ratings were weighted based on individual experience and position. In this study, the following are the weights determined for each member of the team, project manager at 50%, project lead at 20%, and test analysts at 10%.

2.3 Evaluation Criteria

The crisp inputs for each variable were rated based on the previous tasks performed by the testing team. The expert provided the crisp input for each variable and the output is the estimated duration of test script authoring. The results were compared with the actual hours in performing the task and expert-based estimation using the Mean Absolute Relative Error (MARE).

The formula is given as follows:

$$MARE\ \% = \sum_{i=1}^{n} \left\| \frac{(estimate_i - actual_i)}{actual_i} \right\| x\ 100$$

3. Results and Discussion

The tables below show values obtained from the weighted average of the ratings of the experts (Tables 1 to 4). After the second review, the team have agreed on the values shown below. Each row in the tables 1 to 4 consists of parameters of each membership function (category) of each variable. The experts determined (27) twenty-seven fuzzy rules. The surface viewer in Figure 5 and 6 shows all the possible output values of the Scripting Effort Duration based on the inputs for Test Complexity, Test Script Reusability, and Test Scenarios. In Figure 6, as the number of test scenarios and complexity increases the scripting effort duration increases. In Figure 7, as the test script reusability increases the scripting effort decreases when the complexity is low. However, as the complexity increases the scripting effort still increases even though there is high test script reusability.

Table 1. Test Complexity (Scale of 1 to 5)

Category	Smallest	Ideal/Most Promising	Largest
Easy	1.0	1.7	2.4
Average	2.0	2.6	3.5
Complex	3.1	4.0	5.0

Table 2. Test Script Reusability (in percentage)

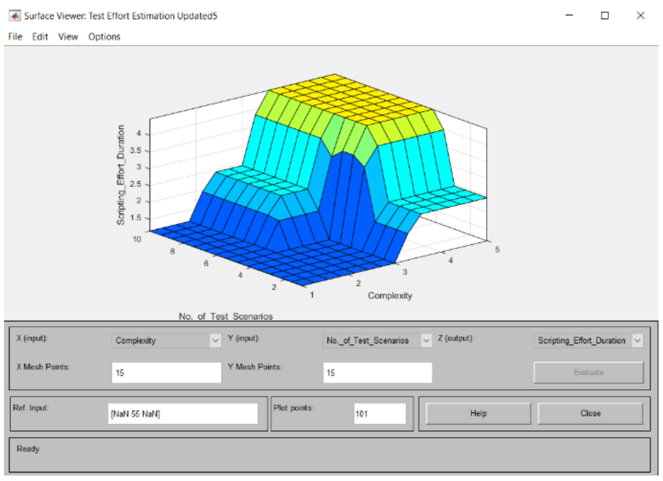
Category	Smallest	Ideal/Most Promising	Largest
Minimal	10	19	29
Partial	29	41	55
Almost	51	71	91 or greater

Table 3. Test Scenarios (Count)

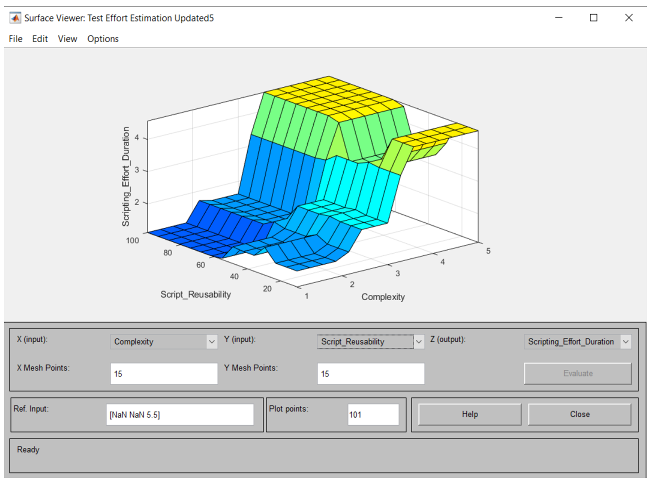
Category	Smallest	Ideal/Most Promising	Largest
Few	1	3	4
Average	3	5	6
Many	5	8	9

Table 4. Test Script Authoring Duration (in hours)

Category	Smallest	Ideal/Most Promising	Largest
Short	0.7	1.0	1.7
Medium	1.6	2.2	3.3
Long	2.8	4.1	5 or greater


Surface Viewer: Test Effort Estimation Updated5
File Edit View Options
Scripting_Effort_Duration
Complexity
No. of Test Scenarios
Evaluate
Help
Close

X (input): Complexity Y (input): No._of_Test_Scenarios Z (output): Scripting_Effort_Duration
X Mesh Points: 15 Y Mesh Points: 15
Ref. Input: [NaN 55 NaN] Plot points: 101
Ready


Surface Viewer: Test Effort Estimation Updated5
File Edit View Options
Scripting_Effort_Duration
Script_Reusability
Complexity
Evaluate
Help
Close

X (input): Complexity Y (input): Script_Reusability Z (output): Scripting_Effort_Duration
X Mesh Points: 15 Y Mesh Points: 15
Ref. Input: [NaN NaN 5.5] Plot points: 101
Ready

Figure 6. Test Scenarios and Test Complexity

Figure 7. Script Reusability and Test Complexity

Table 5 below provides the results from the crisp inputs for each project. The crisp input/rating is in the format of test complexity, test script reusability, and test scenarios count. The crisp input was provided by the project manager. The crisp inputs were inserted in the fuzzy logic model created in MATLAB and the output values were produced. The outputs from the Fuzzy Logic model were shown below and compared with the expert-based estimation. Results showed that the mean absolute relative error of the Fuzzy Logic Model was lower than the currently used Expert-based technique. Figure 8 shows the comparison of the expert-based estimation, fuzzy model estimate and actual test scripting effort duration in hours.

Table 5. Mean Absolute Relative Error (MARE) Comparison of Expert-Based and Fuzzy Logic Model Estimate

Project No.	Expert-Based Estimation (hours)	Fuzzy Logic Model Estimate (hours)	Crisp Input / Rating	Actual Effort (hours)	Mean Absolute Relative Error (MARE) of Expert Based	Mean Absolute Relative Error (MARE) of Fuzzy Logic Model
1	6	4.71	5, 10%, 5	5	20%	-6%
2	5	2.37	5, 40%, 5	4	25%	-41%
3	2	1.13	1, 50%, 1	1	100%	13%
4	2	1.1	3, 40%, 2	1.5	33%	-27%
5	5	2.44	4, 30%, 3	4	25%	-39%
6	4	2.37	5, 10%, 1	3	33%	-21%
7	3	1.1	3, 5% 2	2	50%	-45%
8	6	4.57	5, 20%,4	4	50%	14%
9	2	1.14	2, 80%, 1	1.5	33%	-24%
10	2	1.13	2, 80%, 3	1	100%	13%
Average of MARE					47%	24%

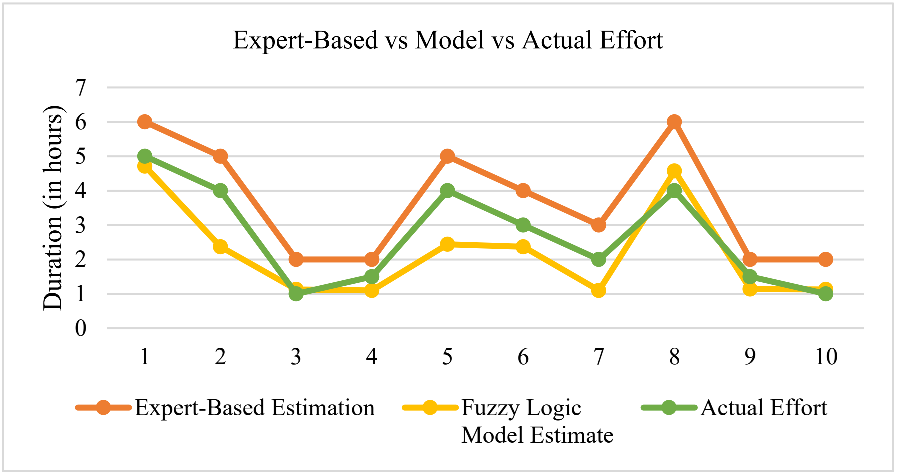


Figure 8. Expert-Based vs Model vs Actual Effort

4. Conclusion and Future Research

This paper illustrates a proposed manual scripting effort estimation tool to a software project using Fuzzy Logic Model with MATLAB. Results demonstrate that fuzzy logic may be used as a decision-support tool in estimating manual test script authoring effort. The analysis of the mean absolute relative error showed that the fuzzy logic model had values nearer to the actual effort duration while the expert-based estimations was higher than the actual estimates. The model used crisp inputs from variables namely Test Scenarios, Test Complexity, and Test Script Reusability to estimate the duration of the scripting effort. This model could support project managers in the test planning phase of the software testing life cycle.

Further studies may incorporate additional input variables tailored to the requirements of each software project. Other factors which affect manual test scripting effort could be added to the existing model. The proposed model could also be compared to other estimation techniques to further validate performance. The ranges and rules of the fuzzy logic model could also be consistently reviewed to update the validity, to improve its accuracy in estimation, and to create a more robust model over a period of time.

References

Abhilasha, & Sharma, A. Test effort estimation in regression testing. 2013 IEEE International Conference In MOOC, Innovation and Technology in Education (MITE). (2013). doi: 10.1109/mite.2013.6756364

Aranha, E., & Borba, P. Estimating Manual Test Execution Effort and Capacity based on Execution Points. International Journal of Computers and Applications, 31(3). (2009). doi: 10.2316/journal.202.2009.3.202-2964

Bluemke, I., & Malanowska, A. Software Testing Effort Estimation and Related Problems. ACM Computing Surveys, 54(3), 1-38. (2021). doi: 10.1145/3442694

Dias-Neto, A., & Travassos, G. A Picture from the Model-Based Testing Area. Advances In Computers, 45-120. (2010). doi: 10.1016/s0065-2458(10)80002-6

Fuzzy Logic (Stanford Encyclopedia of Philosophy). (2022). Retrieved 3 April 2022, from https://plato.stanford.edu/entries/logic-fuzzy/#FuzzLogiVagu

Grover, M., Bhatia, P., & Mittal, H. Estimating Software Test Effort Based on Revised UCP Model Using Fuzzy Technique. Information And Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 1, 490-498. (2017). doi: 10.1007/978-3-319-63673-3_59

Iradier, Á. (2021). Secure software supply chain: why every link matters – Sysdig. Retrieved 1 May 2022, from https://sysdig.com/blog/software-supply-chain-security/

International Software Testing Qualifications Board (ISTQB) (2019). Retrieved 1 May 2022, from https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CTFL_Syllabus_2018_v3.1.1.pdf

Kasurinen, J. Software Organizations and Test Process Development. Advances In Computers, 1-63. (2012). doi: 10.1016/b978-0-12-396526-4.00001-1

Kayacan, E., & Khanesar, M. A. Fundamentals of Type-1 Fuzzy Logic Theory. Fuzzy Neural Networks for Real Time Control Applications, 13–24. (2016). doi:10.1016/b978-0-12-802687-8.00002-5

Malathi S., & Sridhar, S., A Classical Fuzzy Approach for Software Effort Estimation on Machine Learning Technique. (2011).

Mcbride, L. Software Supply Chains: An Introductory Guide. (2021). Retrieved 1 May 2022, from https://blog.sonatype.com/software-supply-chain-a-definition-and-introductory-guide

Mittal, A., Parkash, K., & Mittal, H. Software cost estimation using fuzzy logic. ACM SIGSOFT Software Engineering Notes, 35(1), 1-7. (2010). doi: 10.1145/1668862.1668866

Nisar, M., Yong-Ji Wang, & Elahi, M. Software Development Effort Estimation Using Fuzzy Logic - A Survey. 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery. doi: 10.1109/fskd.2008.370

Saini, A., Ahuja, L., & Khatri, S. K. (2018). Effort Estimation of Agile Development using Fuzzy Logic.7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). 2018 doi:10.1109/icrito.2018.8748381

Sancho-Royo, A., & Verdegay, J. Methods for the Construction of Membership Functions. International Journal of Intelligent Systems, 14(12), 1213-1230. (1999). https://doi.org/10.1002/(sici)1098-111x(199912)14:12<1213: aid-int3>3.0.co;2-5

Sarokolaei, M., Saviz, M., Moradloo, M., & Dahaj, N. Time Driven Activity based Costing by Using Fuzzy Logics. Procedia - Social and Behavioral Sciences, 75, 338-345. (2013). doi: 10.1016/j.sbspro.2013.04.038

Shivakumar, S. Enterprise Web Application Testing. Architecting High Performing, Scalable and Available Enterprise Web Applications, 179-198. (2015). doi: 10.1016/b978-0-12-802258-0.00006-8

Srivastava, P., Kumar, S., Singh, A., & Raghurama, G. Software Testing Effort: An Assessment Through Fuzzy Criteria Approach. Journal Of Uncertain Systems, 5(3), 183-201. (2011). Retrieved from https://www.researchgate.net/publication/235799418_Software_Testing_Effort_An_Assessment_Through_Fuzzy_Criteria_Approach

Taghavifar, H., & Mardani, A. A knowledge based Mamdani fuzzy logic prediction of the motion resistance coefficient in a soil bin facility for clay loam soil. Neural Computing and Applications, 23(S1), 293–302. (2013) .doi:10.1007/s00521-013-1400-4

Tahvili, S., Afzal, W., Saadatmand, M., Bohlin, M., & Ameerjan, S. ESPRET: A tool for execution time estimation of manual test cases. Journal Of Systems and Software, 146, 26-41. (2018). doi: 10.1016/j.jss.2018.09.003

Taley, D., & Pathak, B. Comprehensive Study of Software Testing Techniques and Strategies: A Review. International Journal Of Engineering Research And, V9(08). (2020). doi: 10.17577/ijertv9is080373

Test Scenario: Definition, Purpose, and How to Create - QA Madness. (2022). Retrieved 1 May 2022, from https://www.qamadness.com/knowledge-base/test-scenario-definition-purpose-and-how-to-create/

What Is Fuzzy Logic? (2022). Retrieved 1 May 2022, from https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html.

What is Software Testing and How Does it Work? | IBM. (2022). Retrieved 3 April 2022, from https://www.ibm.com/topics/software-testing

What is Test Scenario? Template with Examples. (2022). Retrieved 1 May 2022, from https://www.guru99.com/test-scenario.html

Wisner, J. D., Tan, K., & Leong, G. K. Principles of Supply Chain Management (5th Edition) (pp. 7,205). Cengage Learning US. (2018). https://bookshelf.vitalsource.com/books/9781337672016

Biography

Ralph Andre C. Roque is a Graduate Student of the School of Industrial Engineering and Engineering Management at Mapua University under the Master of Engineering in Industrial Engineering (MEP-IE) program. He is currently a Test Engineering Analyst in a leading multinational firm providing professional services in digital, cloud, and security. He was formerly a Cadet Pilot for an airline and a Junior Aircraft Material Planner at a German aviation MRO provider in Southeast Asia. He earned his Bachelor of Science in Aeronautical Engineering from Philippine Air Transport and Training Services (PATTS) College of Aeronautics. He is a licensed Aeronautical Engineer since 2018. He is also an active member of the Society of Aerospace Engineers of the Philippines.

Klint Allen A. Mariñas is a Ph.D. student at the Industrial and Systems Engineering Department, Chung Yuan Christian University in Taiwan. He earned his bachelor's degree in Industrial Engineering at Adamson University, Manila, Philippines, and his master's in Industrial Engineering degree at Mapua University, Manila, Philippines. He previously worked as a process engineer in a plastic manufacturing company in the Philippines for three years and eventually took his graduate studies focusing on production planning, human factors, ergonomics, and quality engineering. He is currently under the CIM and Smart Manufacturing laboratory working on ergonomics and production improvement studies.

Michael N. Young is an associate professor in the School of Industrial Engineering and Engineering Management at Mapúa University. He earned his B.S. Industrial Engineering & B.S. Engineering Management from Mapúa Institute of Technology (Philippines) and M.S. & Ph.D. in Industrial and Systems Engineering from Chung Yuan Christian University (Taiwan). His research interests include portfolio optimization and financial engineering.

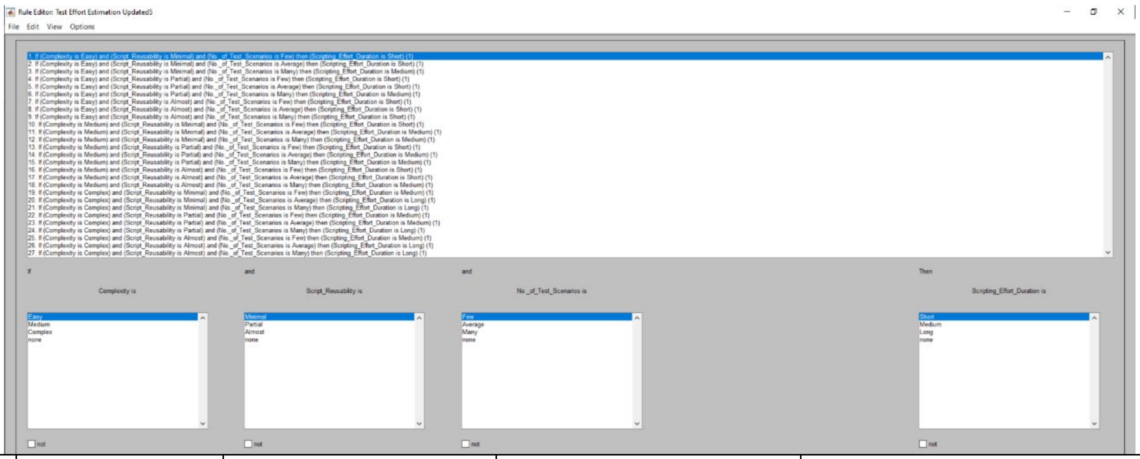
Yogi Tri Prasetyo is an associate professor in the School of Industrial Engineering and Engineering Management, Mapua University, Philippines. He received a B.Eng. in industrial engineering from Universitas Indonesia (2013). He also studied at Waseda University Japan during his junior year (2011-2012) as an undergraduate exchange student. He received an MBA (2015) and a Ph.D. (2019) from the Department of Industrial Management National Taiwan University of Science and Technology (NTUST), with a concentration in human factors and ergonomics. Dr.Prasetyo has a wide range of research interest including color optimization of military camouflage, human-computer interaction particularly related to eye movement, strategic product design, accident analysis, and usability.

Satria Fadil Persada is an Associate Professor/Visiting Professor in the School of Industrial Engineering and Engineering Management, Mapúa University. Dr. Satria has published several journals and conference papers with behavioral science, consumer behavior, and technology acceptance model.

Yung-Tsan Jou received his Ph.D. degree in Integrated (ME, ISE) engineering from Ohio University, Athens, OH, in 2003. He is an Associate Professor of Industrial and Systems Engineering at Chung Yuan Christian University, Taiwan. His research has made contributions in green design, human–system interface design, senior assistive devices, and usability or quality evaluation by using virtual reality tools, smart manufacturing, machine learning, and data analysis.

Annex A

Fuzzy Rules (If-then rule)

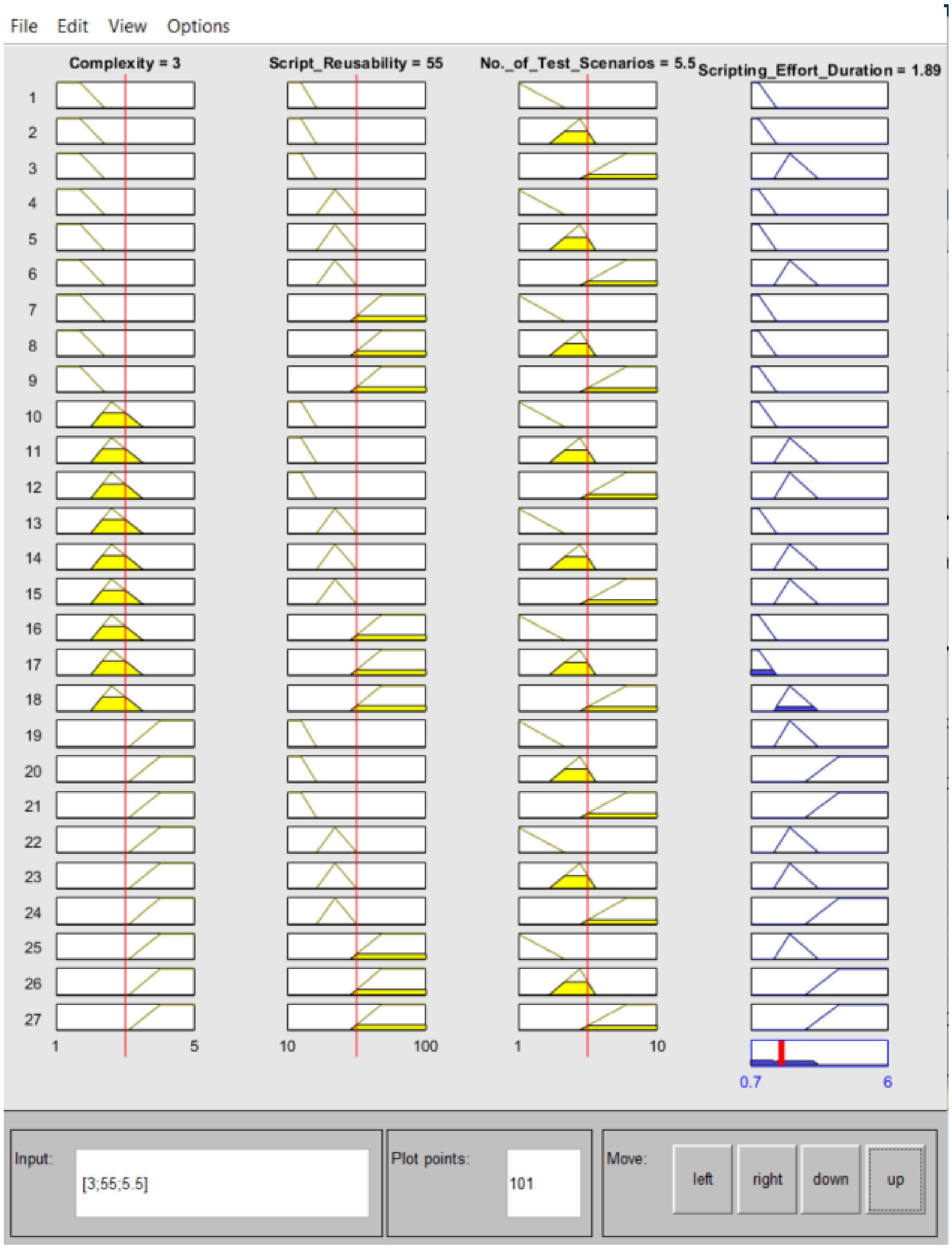


```
1. If (Complexity is Easy) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
2. If (Complexity is Easy) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Short) (1)
3. If (Complexity is Easy) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Medium) (1)
4. If (Complexity is Easy) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
5. If (Complexity is Easy) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Short) (1)
6. If (Complexity is Easy) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Medium) (1)
7. If (Complexity is Easy) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
8. If (Complexity is Easy) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Short) (1)
9. If (Complexity is Easy) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Short) (1)
10. If (Complexity is Medium) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
11. If (Complexity is Medium) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Medium) (1)
12. If (Complexity is Medium) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Medium) (1)
13. If (Complexity is Medium) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
14. If (Complexity is Medium) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Medium) (1)
15. If (Complexity is Medium) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Medium) (1)
16. If (Complexity is Medium) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Short) (1)
17. If (Complexity is Medium) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Short) (1)
18. If (Complexity is Medium) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Medium) (1)
19. If (Complexity is Complex) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Medium) (1)
20. If (Complexity is Complex) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Long) (1)
21. If (Complexity is Complex) and (Script_Reusability is Minimal) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Long) (1)
22. If (Complexity is Complex) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Medium) (1)
23. If (Complexity is Complex) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Medium) (1)
24. If (Complexity is Complex) and (Script_Reusability is Partial) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Long) (1)
25. If (Complexity is Complex) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Few) then (Scripting_Effort_Duration is Medium) (1)
26. If (Complexity is Complex) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Average) then (Scripting_Effort_Duration is Long) (1)
27. If (Complexity is Complex) and (Script_Reusability is Almost) and (No._of_Test_Scenarios is Many) then (Scripting_Effort_Duration is Long) (1)
```

No	If Test Complexity is	and Test Script Reusability is	and No. of Test Scenarios is	Then Scripting Effort Duration is
1	Easy	Minimal	Few	Short
2	Easy	Minimal	Average	Short
3	Easy	Minimal	Many	Medium
4	Easy	Partial	Few	Short
5	Easy	Partial	Average	Short
6	Easy	Partial	Many	Medium
7	Easy	Almost	Few	Short
8	Easy	Almost	Average	Short
9	Easy	Almost	Many	Short
10	Medium	Minimal	Few	Short
11	Medium	Minimal	Average	Medium
12	Medium	Minimal	Many	Medium
13	Medium	Partial	Few	Short
14	Medium	Partial	Average	Medium
15	Medium	Partial	Many	Medium
16	Medium	Almost	Few	Short
17	Medium	Almost	Average	Short
18	Medium	Almost	Many	Medium
19	Complex	Minimal	Few	Medium
20	Complex	Minimal	Average	Long
21	Complex	Minimal	Many	Long
22	Complex	Partial	Few	Medium
23	Complex	Partial	Average	Medium
24	Complex	Partial	Many	Long
25	Complex	Almost	Few	Medium
26	Complex	Almost	Average	Long
27	Complex	Almost	Many	Long

Annex B

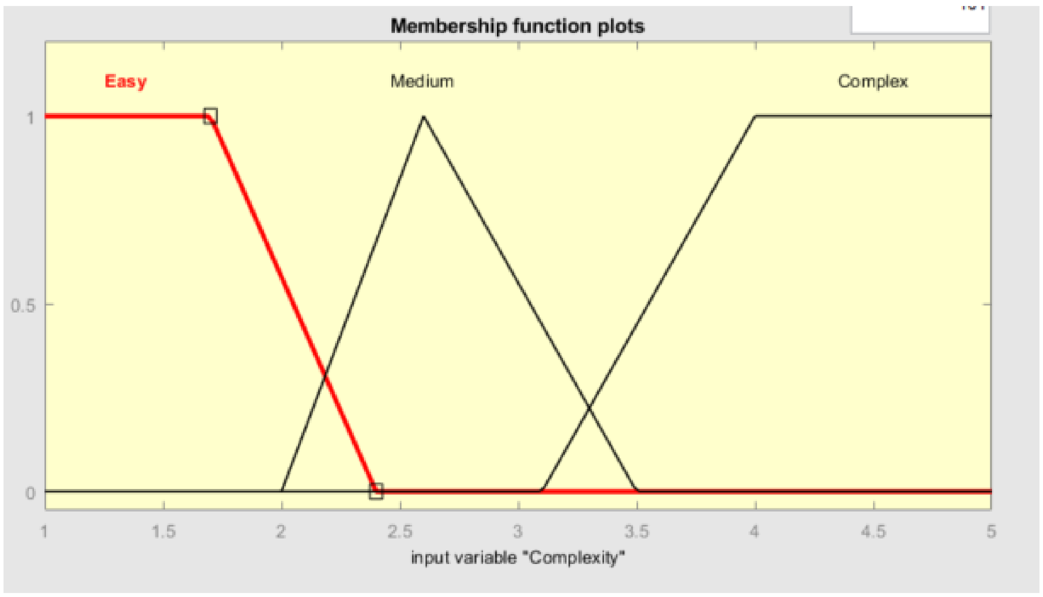
Rule Viewer



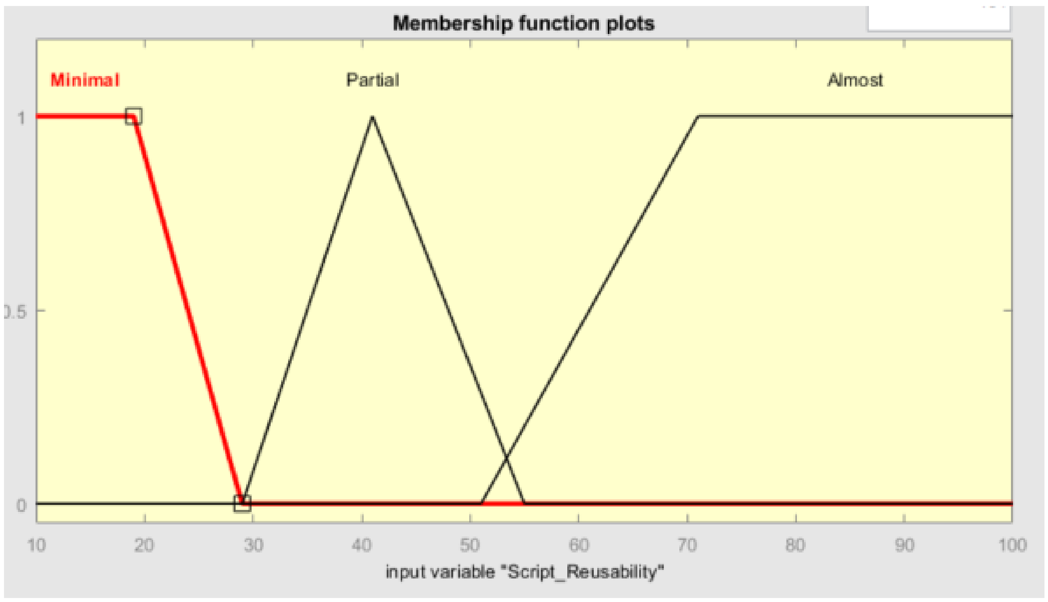
Annex C

Membership Function Plots

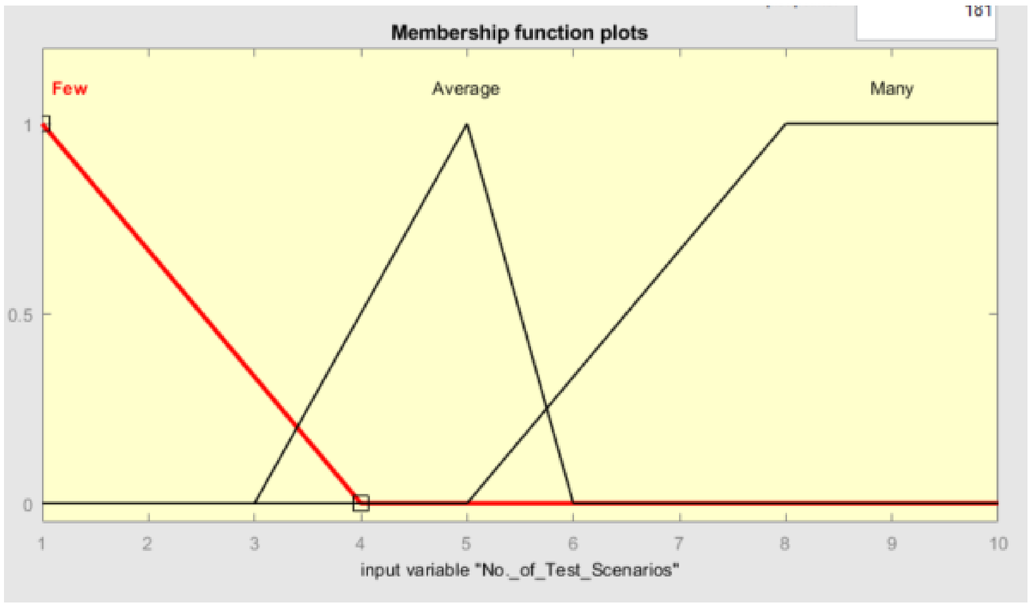
Test Complexity



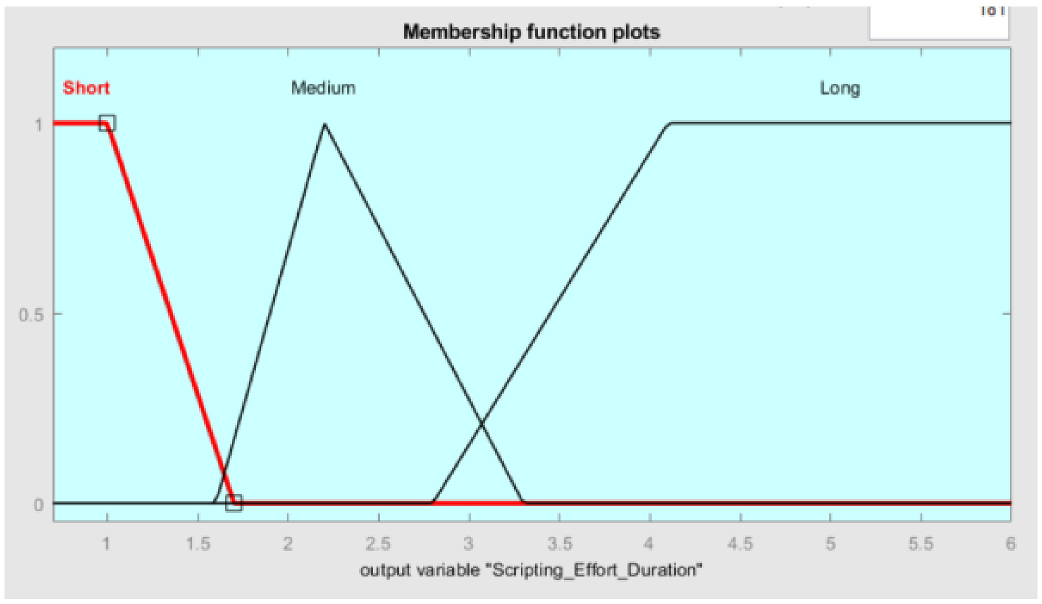
Test Script Reusability



Number of Test Scenarios



Test Script Authoring Duration



Annex D

First Round Survey Results

1. Complexity Rating based on the Demand Requirement/Functionality being tested (Scale of 1 to 5)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Easy	1	1	1	1	1	2	1	1.3	2	1.5	3	1.8	2	2	1.8
Medium	2	1.8	2	2	2	3	2	3	2	2.3	4	3	3.5	3	2.5
Hard	4	2.5	3	3	3	4.5	3	4	4	3.5	5	5	5	5	5

2. Test Script Reusability (in percentage %)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Minimal	5	30	5	5	5	18	35	10	15	10	30	40	20	20	15
Partial	31	30	20	30	20	46	40	30	40	30	60	50	49	50	50
Almost	61	45	40	60	60	81	65	60	70	70	100	80	90	80	80

3. Number of Test Scenarios

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Few	1	2	1	1	1	3	2	2	2	2	4	2	3	2	3
Average	3	2	3	3	4	6	3	4	3	5	7	3	7	5	6
Many	6	3	5	5	7	10	3	8	7	8	11	4	10	8	>10

4. Scripting Effort Duration (in hours)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Short	0.5	1	0.5	1	1	0.8	0.8	1	2	1.5	1	2	2	3	2
Medium	1	2	2	3	2	1.5	2.3	3	4	3	2	3	5	5	4
Long	2	3	3.5	6	5	3	4	5	6	7	>3	4.5	6	>6	>8

Annex E

Second Round Survey Results

1. Complexity Rating based on the Demand Requirement/Functionality being tested (Scale of 1 to 5)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Easy	1	1	1	1	1	2	1	1.3	2	1.5	3	1.8	2	2	1.8
Medium	2	1.8	2	2	2	3	2	3	2	2.3	4	3	3.5	3	2.5
Hard	3.5	2	3	3	3	4.5	3	4	4	3.5	5	5	5	5	5

2. Test Script Reusability (in percentage %)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Minimal	5	30	5	5	5	18	35	10	15	10	30	40	20	20	15
Partial	31	30	20	30	20	46	40	30	40	30	60	50	49	50	50
Almost	51	45	40	60	60	76	65	60	70	70	100	80	90	80	80

3. Number of Test Scenarios

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Few	1	2	1	1	1	3	2	2	2	2	4	2	4	3	4
Average	3	2	3	3	3	6	3	4	3	5	7	3	7	6	6
Many	6	3	5	5	7	10	3	8	7	8	11	4	10	8	10

4. Scripting Effort Duration (in hours)

	Smallest					Ideal / Most Promising					Largest				
	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5	E1	E2	E3	E4	E5
Short	0.5	1	0.5	1	1	0.8	0.8	1	2	1.5	1	2	2.5	3	2
Medium	1	2	1.8	3	2	1.5	2.3	3	4	3	2.5	3	5	5	4
Long	2	2.5	3.5	4	5	3	4	5	6	7	4	4.5	6	7	8