Evaluation of Software Testing Techniques Through Software Testability Index

Ajay Kumar

Assistant Professor's, Ajay Kumar Garg Engineering College, Ghaziabad
Email: ajaygarg100@gmail.com

Abstract : Software testing is one of the most important phases in the Software Development Life Cycle (SDLC) which helps in assessing the quality of the software. There are a number of testing techniques at various phases of testing. Selection of the right testing technique at any stage is one of the critical problems and depends on many factors such as resources, schedule, cost of the project, etc. The appropriate usage of efficient testing technique at various stages of the SDLC is still in infant stage. There is a need for a selection method that not only takes the subjective knowledge but also incorporates the objective knowledge for the efficient selection of testing technique as per the requirements. The paper proposed a method to calculate Software Testability Index of software systems which can be used to evaluate the testing techniques. The method formulates the problem as a Multi Criteria Decision Making problem and the solution is based on Technique for Order Preference by Similarity to Ideal Solution(TOPSIS) and Analytic Hierarchy Process(AHP).

General Terms : Testing, Unit Testing, Integration Testing, Functional Testing, Control Flow Testing, Data Flow Testing, Software Testability Index and Evaluation Criteria.

Keywords : Multi criteria decision making approach, Testing and Software Testability Index.

I. INTRODUCTION

Software is the set of programs along with the associated documentation and configuration data needed for the proper functioning of the correct operation [17]. Some of the potential areas where software can be used are Systems software, Real time applications, Business information processing, Engineering and scientific applications, embedded applications, web based application etc. [12]. A software development process also known as Software Development Life Cycle (SDLC) is a set of activities that leads to the production of a software product.

II. RELATED WORK

The research done in the area of software testing includes improving the effectiveness of testing based on reducing the number of test cases, experimental evaluation of testing techniques, based on data flow relationship, fault coverage, prioritizing the test cases based on certain objective function, rate at which faults are detected, etc.

Basili, Selby (1987) has done an evaluation of software testing techniques where they compared code inspection with functional and structural testing techniques[1]. Rothermal, et.al., (1996) uses a comparison framework for regression testing techniques[14].

Mahapatra, et.al., (2008) proposed a technique for improving the efficiency of software testing, which is based on a conventional attempt to reduce test cases that have to be tested for any given software. The approach utilizes the advantage of regression testing where fewer test cases would lessen time consumption of the testing as a whole [9].

Rothermel, et.al., (1999) proposes that test case prioritization techniques to schedule test cases for execution in an order that attempts to maximize some objective function[15]. One such function involves rate of fault detection — a measure of how quickly faults are detected within the testing process.

Abhijit A. Sawant, Pranit H. Bari, P. M. Chawan (2012) describes in detail about software testing, need of software testing, Software testing goals and principles. some of the strategies supporting these paradigms, and have discussed their pros and cons [7].

Shivkumar Hasmukhrai Trivedi (2012) describe that test automation reduces testing costs by supporting the test process with a range of software tools [3]. One of the major system problems within software testing area is how to get a suitable set of cases to test a software. This paper focus on the importance of test cases according to them test cases are most important elements of software testing. if test cases are designed in better manner then we can result with better software testing of a software product.

Kamala Ramasubramani Jayakumar, Alain Abran (2013) presents an overview of software test estimation techniques surveyed, as well as some of the challenges that need to be overcome if the foundations of these software testing estimation techniques are to be improved [4].

Milad Hanna, Nahla El-Haggar, Mostafa Sami (2014) aims to compare the main features of different scripting techniques used in process of automating the execution phase in software testing process. According to them

automation of testing phase offers a potential source of savings across all the life cycle [5].

These methods aims at improving the testing scenario considering few factors at a time. The work done so far in the area shows the need for a formulized method to help in selecting the best technique at various stages of testing.

III. TESTING TECHNIQUES

Testing is involved in various phases of the software development life cycle with different objectives. It can broadly be classified as static and dynamic testing.

1.1 Unit Testing

Unit testing is testing program units in isolation. This is the lowest level of testing performed during software development. A unit is usually a function or a procedure or a method [10].

The cost in terms of effort required for preparing test cases, executing the test cases, analyzing the results and fixing the defects is very less in this stage compared to other stages of testing.

1.1.1 Control Flow Testing

Control flow testing uses the control structure of the program as the basis for developing test cases. Some methods used for selecting paths in control flow testing are:Segment or Statement testing, branch testing, and condition testing, modified condition-decision coverage, path testing, structured path testing, and boundary-interior path testing [2].

1.1.2 Data Flow Testing

Data flow testing looks at interactions involving definitions of program variables to subsequent references that are affected by the definitions.

1.2 Integration Testing

The main goal of this testing is to build a working version of the system by putting modules together in an incremental manner and ensuring that additional modules work as expected without disturbing functionality of modules already put together. This helps in finding the interface errors between the modules [10]. Test cases are explicitly written so that they examine the interfaces between the various units.

The unit modules are usually developed by people of different groups. So when two modules are integrated there is a chance for interface errors to creep into the system even though we take effort in designing the system properly.

1.3 Functional Testing

This is also known as behavioral testing. Test cases are set of inputs which will test all the functional requirements of the system under test. It is useful in finding incorrect or missing functions, interface errors, errors in Data structures, DB Access, behavior/performance errors; initialization and termination errors . and execute the system and analyze if the output is present in the output domain [10].

The different functional techniques are boundary value analysis, equivalence partition, comparison testing, random testing, adaptive random testing, error guessing.

IV. EVALUATION CRITERION

After discussing the testing techniques, the next step is to come up with the criteria that can be used to evaluate the testing techniques.

1.4 Criteria

A few classes of criteria were identified to evaluate the testing techniques. They can be broadly categorized as: General details of the technique, Data Source, Effort, Training/Learning, Coverage, Resources, Organizational details, Project details, Types of defects, Automation properties, Types of Project. A value NA in any column indicates Not Applicable.

Table 4.1 gives the criteria which fall under the effort category. This includes effort required for test case generation, test case execution, and additional work needed to be done before starting the testing.

Table 4.1 Effort related criteria to evaluate testing techniques

Attributes	Qualitative	Quantitative
Test case generation effort – manual	2	NA
Test case execution effort – manual	2	NA
Test case generation effort – automation	2	NA
Test case execution effort – automation	2	NA
Additional Work	4	NA

Table 4.2 discusses the criteria falling under the training/learning class. Testers need to be given training so that they understand the usage of the technique. The following criteria can be used to evaluate the time taken for that.

Table 4.2 Training/Learning related criteria to evaluate testing techniques

Attributes	Qualitative	Quantitative
Learning time	2	NA
Training available	4	NA
Training for automation tool	4	NA

Table 4.3 considers various coverage criteria used as metrics to find the effectiveness of testing.

Table 4.3 Coverage criteria

Attributes	Qualitative	Quantitative
Requirement	4	NA
Design	3	NA
Code	2	NA

Table 4.4 gives details of the criteria which can be used as a source for generating the test cases while testing any system.

Table 4.4Resources for generating test cases

Attributes	Qualitative	Quantitative
Requirement	4	NA
Design	3	NA
Code	2	NA

V. SOFTWARE TESTABILITY INDEX

Testing techniques used in various stages of software testing and the criteria used to evaluate them have been discussed. Now we focus on, the method which are used to solve the problem of evaluation and selection of testing techniques. Software Testability Index (STI) is calculated and is used as a measure to select the best testing technique. The method formulates the problem as a multi criteria decision making problem and proposes a solution.

1.5 MCDM: Multi Criteria Decision Making Approach.

This section discusses a solution to solve the selection of a testing technique problem framing it as a Multi Criteria Decision Making(MCDM) problem. The method uses TOPSIS along with AHP [6] in solving the problem. TOPSIS is based on the idea that the best alternative has the shortest distance from the positive ideal solution and is the farthest from the negative ideal solution [6]. AHP is used to calculate the weights of the criteria used for evaluation [16]. AHP uses consistency ratio as a measure to verify the consistency of the weights obtained for the criteria. This ensures that the weights of the criteria are consistent and can be used further. In this method, relative closeness from the positive and negative ideal solutions is calculated and is used as the testability index to rank the testing techniques.

The following steps need to be performed to choose the best alternative when n criteria are used to compare m alternatives.

Step 1 - Decision Matrix $\mathbf{[D]}_{m \times n}$

In this step, a decision matrix of criteria and alternatives based on the available information is developed. The value d_{ij} of the matrix is the actual value of the ith alternative in terms of jth decision criteria i.e., the rating of the ith alternative with respect to the jth criteria. The order of the decision matrix is m × n.

$$Dmxn = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \dots & \dots & \dots & \dots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{pmatrix} \quad (5.1)$$

Step 2 - Normalized decision Matrix $\mathbf{[R]}_{m \times n}$

The decision matrix **[D]** is normalized so that ratings given in different scales and units to various criteria are transformed into common measurable units to make comparisons easy. The order of the normalized decision matrix is m × n. The value r_{ij} of the matrix **[R]** is calculated using equation (5.2).

$$r_{ij} = d_{ij} / [\sum_{i=1}^{m} (d_{ij})^2]^{0.5} \quad (5.2)$$

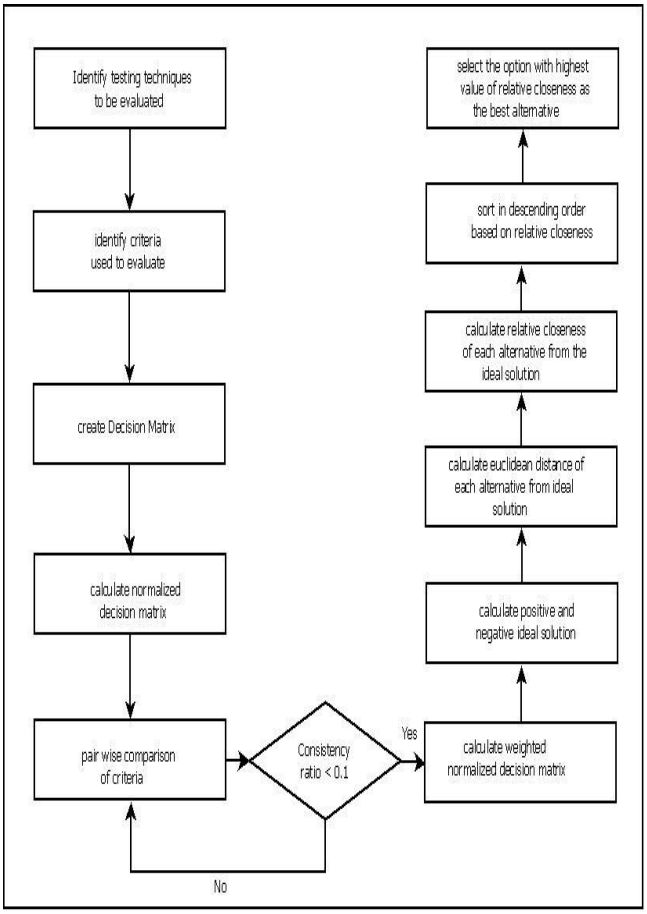

Figure 5.1depicts the flow chart of the methodology proposed.

Step 3 – Attribute Weights $\mathbf{[W]}_{1\times n}$

AHP [16] is used in this step to calculate the weights of the criteria used to evaluate the alternatives. A pair wise comparison of the criteria is done and the absolute weights of the criteria are calculated using AHP.The pair wise comparison is done on a scale of 1-9 and the values along with its explanation are shown in detail in table below.

Table 5.1AHP rating scale

Value	Explanation
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very Strong importance
9	Extreme importance
2,4,6,8	Intermediate values

If A is the pairwise comparison matrix, the relative weights of the criteria **[W]** is calculated using the equation (5.3).

$$(A - \lambda_{max} I) \times W = 0 \qquad (5.3)$$

whereλ_{max} is the largest Eigen value of the matrix A and I is the n × n unit matrix.

The value of consistency ratio while calculating the weights is considered as a factor which judges the consistency of the absolute weights given to the criteria. Any value less than 0.1 implies that the judgment is consistent. The consistency ratio is calculated using the equation (5.4).

$$CR = CI / RI \qquad (5.4)$$

where CI is consistency index and RI is the random consistency index. The value of the consistency index is calculated using the equation (5.5).

$$CI = (\lambda_{max} - N) / (N - 1) \qquad (5.5)$$

where N is the number of criteria.

The values of the random consistency index for different values of N are given in Table 5.2.

Table. 5.2 Random consistency index

N	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.9	1.12	1.24	1.3	1.41	1.45

Step 4 – Weighted Normalized Matrix **[V]**m×n

The weighted normalized matrix **[V]** is calculated by multiplying each column of the normalized decision matrix **[R]** with the respective column of the **[W]** matrix. The value v_{ij} of the matrix **[V]** is calculated using equation (5.6).

$$v_{ij} = W_j \times r_{ij} \qquad (5.6)$$

Step 5 – Positive ideal solution $\mathbf{[I_s^+]}_{n\times 1}$ and Negative ideal solution $\mathbf{[I_s^-]}_{n\times 1}$

The positive ideal solution is calculated as the best value each criterion can attain. The negative ideal solution is calculated as the least/worst value each criterion can attain. They are produced as follows using the equation 5.7 and 5.8.

$I_s^+ = \{ (\max v_{ij} / j \in \tau), (\min v_{ij} / j \in \tau') \text{ for } i=1,2,3,\ldots,m\}$

$= \{V_1^+, V_2^+, \ldots, V_n^+\}$ (5.7)

$I_s^- = \{ (\min v_{ij} / j \in \tau), (\max v_{ij} / j \in \tau') \text{ for } i=1,2,3,\ldots,m\}$

$= \{V_1^-, V_2^-, \ldots, V_n^-\}$ (5.8)

where τ is associated with the benefit or positive criteria and τ' is associated with the cost or negative criteria.

Step 6 – N dimensional Euclidean distance – $\mathbf{[S_i^+]}_{m\times 1}$ and $\mathbf{[S_i^-]}_{m\times 1}$

The N dimensional Euclidean distance is calculated from each alternative to the positive and negative ideal solution using the equations 4.9 and 4.10.

$$S_i^+ = \{\sum_{j=1}^{n} (V_{ij} - V_j^+)\}^{0.5}, \; i = 1,2,\ldots,m \qquad (5.9)$$

and

$$S_i^- = \{\sum_{j=1}^{n} (V_{ij} - V_j^-)\}^{0.5}, \; i = 1,2,\ldots,m \qquad (5.10)$$

Step 7 – Relative Closeness $\mathbf{[C_i^*]}_{m\times 1}$

Relative closeness of each alternative with respect to the positive and negative ideal solution is calculated using the equation 5.11.

$$C_i^* = S_i^- / (S_i^+ + S_i^-) \qquad (5.11)$$

Step 8 – Ranking

The alternatives are arranged in the descending order of their relative closeness (C^*) from the ideal solutions. The alternative with the maximum value of C^* is chosen as the best alternative.

VI. RESULTS AND DISCUSSION

The integration testing techniques identified for testing the software are A = {Incremental Integration, Top-Down integration testing, Bottom-up integration testing, Sandwich, Big bang, End-to-End testing, High Frequency testing, Smoke testing}.

The set of criteria considered for evaluating and selecting the best alternative is C = {Time when the basic functionality is exposed, Reusability of the test cases, End user view, Time of fault detection, Effort required in terms of the additional work to be done, Ease of fixing errors, Frequency of running the tests, Ease of writing the test cases, and Possibility of automating the testing}.

1.6 Evaluation of Functional Testing Techniques

The functional testing techniques identified for testing the evaluation are A = {Boundary value analysis, Equivalence partitioning, Comparison testing, Random testing, Error guessing}.

The set of criteria considered for evaluating and selecting the best alternative is C = {Domain errors, Expertise needed, End user view, Execution effort}.

The decision matrix created for evaluating functional testing techniques is shown in Table 6.1.

Table 6.1 Functional Testing: Decision Matrix

	C1	C2	C3	C4
A1	5	4	3	5
A2	5	4	3	5
A3	3	3	2	4
A4	2	2	4	2
A5	3	1	4	2

The decision matrix after normalization is shown in Table 6.2.

Table 6.2 Functional Testing: Normalized Decision Matrix

	C1	C2	C3	C4
A1	0.589256	0.589768	0.408248	0.581238
A2	0.589256	0.589768	0.408248	0.581238
A3	0.353553	0.442326	0.272166	0.464991
A4	0.235702	0.294884	0.544331	0.232495
A5	0.353553	0.147442	0.544331	0.232495

The weight matrix obtained after using AHP is shown in Table 6.3. The consistency ratio obtained while calculating the weights of the criteria is 0.07 which is less than 0.1 and hence the weights are considered consistent.

Table 6.3 Functional Testing: Weight Matrix

Criteria	C1	C2	C3	C4
Weight	0.082800	0.210800	0.149600	0.556900

The weighted normalized decision matrix is shown in Table 6.4

Table 6.4 Functional Testing: Weighted Normalized Decision Matrix

	C1	C2	C3	C4
A1	0.048790	0.124323	0.061074	0.323692
A2	0.048790	0.124323	0.061074	0.323692
A3	0.029274	0.093242	0.040716	0.258953
A4	0.019516	0.062162	0.081432	0.129477
A5	0.029274	0.031081	0.081432	0.129477

The positive and negative ideal solutions for each criterion are shown in Table 6.5.

Table 6.5 Functional Testing: Positive and Negative Ideal solution

	C1	C2	C3	C4
Is+	0.048790	0.124323	0.081432	0.323692
Is-	0.019516	0.031081	0.040716	0.129477

The values of the N-dimensional Euclidean distance from the positive and negative ideal solutions and Relative closeness C^* for each of the alternative are shown in Table 6.6.

Table 6.6 Functional Testing: Euclidean distance, relative closeness and ranking

	A1	A2	A3	A4	A5
S+	0.1426	0.1426	0.39503	0.53446	0.55405
S-	0.5805	0.5805	0.44877	0.26794	0.22466
C*	0.8027	0.8027	0.53184	0.33393	0.28850
Rank	1	1	3	4	5

Table 6.8 shows the values of software testability index and the ranking of the Functional testing techniques.

Table 6.8 Functional Testing: Testability Index

Testing Technique	A1	A2	A3	A4	A5
Rank	1	1	3	4	5

Figure 6.1.1 show the histogram of the values obtained using MCDM for Functional Testing.

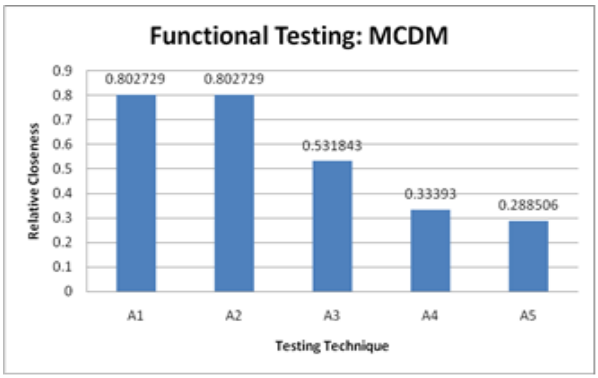

Fig. 6.1.1 Functional Testing: MCDM

VII. CONCLUSION

Testing is done at various phases of the SDLC with different objectives. The code written to implement the software is unit tested. It looks at the interactions involving the variables and control flow of the module. The unit tested modules or components are then integrated in any of the standard methods of integration and the integration between the unit tested modules are checked. Functional testing is done on the system to find out any missing or incorrect functions. There are various testing techniques for each of the above mentioned levels of testing. A methodology which can be used to select a testing technique based on some evaluation criteria, irrespective of the phase of testing is proposed.

Firstly the testing techniques used at various phases of the SDLC are identified and knowledge hub is established. Next the criteria that is used to compare, evaluate and select the best possible testing technique is identified. The criteria identified can be broadly

classified as general details of the testing techniques, the effort required, training / learning required, coverage, resources required, organizational details, project details, types of defects and the possibility of automation.

One method of evaluating the software testing techniques is explained. The methods use an index called Software Testability Index to evaluate and rank the testing techniques. The proposed method formulates the selection problem as a Multi Criteria Decision Making problem and uses TOPSIS along with AHP in choosing the best alternative. TOPSIS works on the principle that the best alternative is at a maximum distance from the negative ideal solution and at a minimum distance from the positive ideal solution. The method uses relative closeness to the ideal solution as a measure of identifying the best alternative. AHP is used in finding the weights of the criteria. AHP uses consistency ratio as a measure to check if the weights obtained are consistent.

This method can be used in any software organization for the right selection of testing technique at any stage of the SDLC. Criteria like resources required (human or computational), previous use of a testing technique in the organization, ease of fixing the defects by the developers, training required before the use of the technique by a tester can also be used by the testing team according to the schedule, cost or resources requirement of the project in the process of decision making. Thus the method considers not only the subjective knowledge but also practical aspects of the testing techniques in choosing the best testing technique to be used.

REFERENCES

[1] Basili, V.R., Selby, R.W. , 'Comparing the effectiveness of software testing'. IEEE Transactions on Software Engineering.vol SE-13 No. 12. pp 1278–1296,(1987).

[2] Beizer, B. ,'Software Testing Techniques'. Dreamtech press pp. 145-172,(2003).

[3] Shivkumar Hasmukhrai Trivedi,' Software Testing Techniques', International Journal of Advanced Research in Computer Science and Software Engineering, Vol 2,pp 433-438, (2012).

[4] Kamala Ramasubramani Jayakumar, Alain Abran, 'A Survey of Software Test Estimation Techniques ',Journal of Software Engineering and Applications, vol 6, pp. 47-52,(2013).

[5] Milad Hanna, Nahla Haggar, Mostafa Sami 'A Review of Scripting Techniques used in Automated Software Testing', International Journal of Advanced Computer Science and Applications, Vol. 5, No. 1, (2014).

[6] Jadidi, O., Hong, T.S., Firouzi, F., Yusuff, R.M., Zulkifli, N. ,'TOPSIS and fuzzy multi-objective model integration for supplier selection problem' Journal of Achievements in Materials and Manufacturing Engineering.vol 31 No. 2. pp 762 – 769, (2008).

[7] Abhijit A. Sawant, Pranit H. Bari, P. M. Chawan, 'Software Testing Techniques and Strategies' ,International Journal of Engineering Research and Applications, Vol. 2, issue 3, pp.980-986, (2012).

[8] Ma, Z., Zhao, J. ,'Test case prioritization based on analysis of program structure'. Asia-Pacific Software Engineering Conference. pp 471 – 478, (2008).

[9] Mahapatra, R.P., Singh, J. , 'Improving the Effectiveness of Software Testing through Test Case Reduction'. World Academy of Science, Engineering and Technology 37. pp 345-350, (2008).

[10] Naik, K., Tripathy, P. , 'Software Testing and Quality Assurance'. Wiley, pp 51-250, (2008).

[11] Ntafos, S.C. ,'A comparision of some structural testing strategies'. IEEE Transactions on software engineering vol 14 No. 6., pp 868-874, (1988).

[12] Pressman, R.S. ,'Software Engineering: A practitioner's approach'. McGraw-Hill pp. 437-503, (2001).

[13] Rajendra ,R.V.,'White paper on unit testing',http://www.mobilein.com/WhitePaperon UnitTesting

[14] Rothermel, G., Harrold, M.J. ,'Analyzing Regression Test Selection Techniques', IEEE Transactions On Software Engineering, Vol. 22, No. 8., pp 529-551, (1996).

[15] Rothermel, G., Untch, R.H., Chu, C., Harrold,M.J. , 'Test Case Prioritization: An Empirical Study'. Proceedings of the International Conference on Software Maintenance. pp 179 – 188, (1999).

[16] Saaty, T.L. ,'The Analytic Network Process', RWS Publications, (1996).

[17] Sommerville, I. ,'Software Engineering'. Pearson Education, (2009).