BW-TOPSIS: A Hybrid Method to Evaluate Software Testing Techniques

Ajay Kumar, and Kamaldeep Kaur

Original scientific article

Abstract—**Software testing plays a significant role in various software development phases. There are so many software testing techniques available. Selecting the most suitable software testing technique based on multiple factors is challenging for software practitioners. This paper proposes an MCDM-based hybrid approach for selecting the most appropriate software testing technique among various available software testing techniques, considering multiple factors such as cost, schedule, resources, etc. Because of the involvement of multiple factors, the problem of selecting the most appropriate software testing technique can be modeled as an MCDM problem. This study proposes a hybrid approach by integrating two MCDM methods BWM (Best-Worst Method) and TOPSIS (Technique for Order Preference by Similarity to Ideal Solution), for evaluating various software testing techniques considering multiple factors altogether. For the applicability of the proposed approach, an experimental study was conducted using seven software testing techniques and six evaluation criteria. Results show the proposed approach can be used as an efficient tool for selecting the most suitable software testing technique among various available testing techniques in the presence of multiple factors.**

Index terms— **Software Testing Technique; Multi-Criteria Decision Making; BWM; TOPSIS.**

I. INTRODUCTION

In day-to-day life, software plays a vital role in many applications, such as home appliances, industrial controls, hospital health care units, nuclear reactor plants, aircraft, air traffic control, shopping, and many more. To increase their effectiveness and efficiency, many governments and commercial organizations depend on the proper functioning of the software. Software failure may lead to economic loss and customer dissatisfaction for the organizations. So, in this scenario, assessing the quality of the software is an essential task.

Software testing is a significant sub-space of software quality affirmation, which assists software practitioners with discovering bugs and mistakes during software product development. Software testing is a necessary task which is to be needed at various stages in the development of any software to ensure the proper functioning of software according to the specific requirements [1]. There are different software testing techniques available but selecting the most suitable testing technique is a crucial task because selecting the right testing technique depends upon various factors such as cost, schedule, resources, etc.

MCDM is a technique that is used to select the most appropriate alternative from the different available alternatives to solve a particular problem in the presence of various conflicting criteria [2]. In this paper, the problem of selecting the most suitable software testing technique from different available software testing techniques (alternatives) can be modeled as an MCDM problem since the selection of the right software testing technique involves more than one factor (criteria).

This paper proposes a hybrid approach by integrating two MCDM methods, BWM and TOPSIS, for evaluating various software testing techniques taking various factors into consideration. For the validation of the proposed approach, we have conducted an experimental study using seven integration testing techniques considering six different factors as evaluation criteria. To the best of our knowledge, no previous research has attempted to evaluate software testing techniques using an MCDM-based hybrid approach by integrating BWM and TOPSIS, considering various factors altogether. The proposed MCDM-based hybrid approach will be useful in aiding software practitioners in selecting the most suitable software testing technique during the various phases of software development.

The remaining part of this study is as follows. Section II presents research done related to the evaluation of software testing techniques. Section III presents the proposed MCDM-based hybrid approach. Section IV presents the experimental study to validate the proposed approach. Section V discusses the results, section VI highlights the theoretical and practical implications, and section VII concludes the paper.

II. RELATED WORK

Vos et al. [3] proposed a framework for the evaluation of software testing techniques. According to them, this framework

Manuscript received October 25, 2022; revised November 23, 2022. Date of publication December 6, 2022. Date of current version December 6, 2022. The associate editor prof. Darko Huljenić has been coordinating the review of this manuscript and approved it for publication.

A. Kumar is with the Department of Information Technology, KIET Group of Institutions, India. K. Kaur is with the University School of Information, Communication & Technology (USIC&T), Guru Gobind Singh Indraprastha University, India. E-mails: ajaygarg100@gmail.com, kdkaur99@ipu.ac.in.

Digital Object Identifier (DOI): 10.24138/jcomss-2022-0138

will be helpful for software engineers in defining test cases. Babu et al. [4] conducted a systematic review of software testing strategies on the basis of the evaluation and classification of software testing techniques. Lemos et al. [5] conducted a review of evaluation studies in software testing research and emphasized the importance of the evaluation of software testing techniques. Neto and Travassos [6] proposed a method based on the combined selection of model-based software testing methods for using software testing techniques effectively. Atifi et al. [7] presented a comparative study of two software testing techniques, namely risk-based testing (RBT) and model-based testing (MBT). Dallal et al. [8] presented a comparative analysis of various software testing techniques specifically used for aspect-oriented software systems.

Ibarra and Rodriguez [9] proposed a content-based system for the evaluation of software testing techniques based on the characteristics of a target project using a collaborative approach. Shuaibu et al. [10] investigated different types of software testing techniques by comparative analysis based on different characteristics of the software system. Farooq [11] presented an empirical study for the assessment of three software testing methods, namely functional, code reading, and structural testing.

Khari and Kumar [12] presented a survey of research work on the evaluation of search-based software testing techniques between 1996 to 2016. Sharma et al. [13] analyzed the impact of ontology on software testing. The authors also discuss the factors affecting software testing directly or indirectly. Qasim et al. [14] conducted a systematic literature review of test case prioritization methods in regression testing. Martensson et al. [15] proposed a tangible model for the efficient testing of large-scale software products.

Ali et al. [16] proposed a model for prioritizing and selecting a test case to improve the quality of a software release. Their proposed model follows two steps for prioritizing and selecting the test cases. First, the most frequently changing test cases are clustered together and prioritized. Second, test cases are selected based on the higher failure rate. The authors conducted an experimental study considering three industrial software projects to compare their proposed approach with existing regression techniques. The results show that the proposed model outperforms alternative regression techniques.

Juhnke et al. [17] performed a case study to identify the problems related to test case specifications in the automation of software testing. The identified problems were summarized in nine categories: availability, content-related problems, quality assurance, tools, communication, processes, test case specification content, test case description, and lack of knowledge. The authors emphasized the necessity of quality assurance measures for specifying test cases. Abusalim et al. [18] evaluated software testing techniques were focusing on mobile application software systems.

Beyer [19] reported the results of the first international competition on software testing. The author concludes that the new standards for making input values, writing the generated test suites, and specifying test coverage criteria will encourage test case generators to apply them for delivering the testing tools that can be used easily as quality assurance components. Jung et al. [20] developed a model to reduce redundant test executions during the testing of a software project.

Following a thorough review of related work, it is clear that the majority of previous research has focused on improving the effectiveness of software testing techniques. However, some researchers have worked towards the evaluation of software testing techniques, but they have considered only one evaluation criterion at a time. As the selection of software testing techniques depends upon various characteristics (evaluation criteria) of the project, it is essential need to develop a framework that can be used to evaluate software testing techniques in the presence of more than one evaluation criterion. This paper proposes a hybrid approach by integrating two MCDM methods, BWM and TOPSIS, for evaluating various software testing techniques taking various factors into consideration altogether.

III. Proposed Method

The problem of selecting the most suitable software testing technique can be modeled as an MCDM problem since the performance of software testing techniques (alternatives) may depend upon various factors (evaluation criteria) such as cost, schedule, resources, etc. This study proposes a hybrid approach BW-TOPSIS by integrating two MCDM methods, BWM and TOPSIS, for the selection of the most suitable software testing techniques from different available software testing techniques (alternatives) considering various factors (evaluation criteria).

The proposed method uses the concept of TOPSIS [21] along with BWM [22] to select the most appropriate software testing technique. TOPSIS selects the best alternative based on the distance of the alternative from the positive ideal solution and negative ideal solution. The alternative which has the farthest distance from a negative ideal solution and the shortest distance from a positive ideal solution is considered the best alternative. Weights of the evaluation criteria are calculated by using BWM. A graphical representation of the proposed methodology is shown in Fig. 1.

The detailed stepwise procedure of the proposed BW-TOPSIS hybrid approach for evaluating *m* alternatives in the presence of *n* criteria is given below.

Step1: Construction of Decision matrix $[M]_{m\times n}$

First, a decision matrix $[M]_{m\times n}$ is constructed as shown in "(1)" in which each value of M_{ij} represents the performance of i^{th} alternative with respect to j^{th} criterion.

$$M_{m\times n} = \begin{pmatrix} M_{11} & M_{12} & M_{13} & \cdots & M_{1n} \\ M_{21} & M_{22} & M_{23} & \cdots & M_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & M_{m3} & \cdots & M_{mn} \end{pmatrix} \quad (1)$$

Step2: Construction of Normalized Decision matrix $[NM]_{m\times n}$

The decision matrix obtained from the previous step is normalized to convert ratings given in various scales and units of criteria into a single measurable unit. The value of NM_{ij} of normalized decision matrix $[NM]_{m\times n}$ can be calculated by using the following equation.

$$NM_{ij} = \frac{M_{ij}}{\sqrt{\sum_{i=1}^{m} M_{ij}^{2}}} \tag{2}$$

Step3: Construction of Criteria Weight Matrix $[CW]_{1\times n}$

The MCDM method Best-Worst method (BWM) [22] is used to calculate the absolute weights of all the criteria. In this MCDM method, the most desirable criterion is taken as the best criterion, and the least desirable criterion is taken as the worst criterion. These two criteria (best and worst) are then pairwise compared with other criteria. The weights of various criteria are then determined by formulating and solving a maximin problem. The detailed procedure to calculate the weight matrix by using BWM is given below.

Step3(a): Consider the n number of criteria for which the weight matrix is to be calculated.

Step3(b): Choose the most desirable criterion as the best criterion and the least desirable criterion as the worst criterion.

Step3(c): Find the preference of the best criterion over other criteria using a number scale from 1 to 9. These preferences are represented in the form of a best-to-others vector as follows.

$$C_B = (c_{B1}, c_{B2}, \ldots.., c_{Bn}) \tag{3}$$

where c_{BJ} denotes the preference of the best criterion B with respect to criterion j. It may be noted that $c_{BB} = 1$.

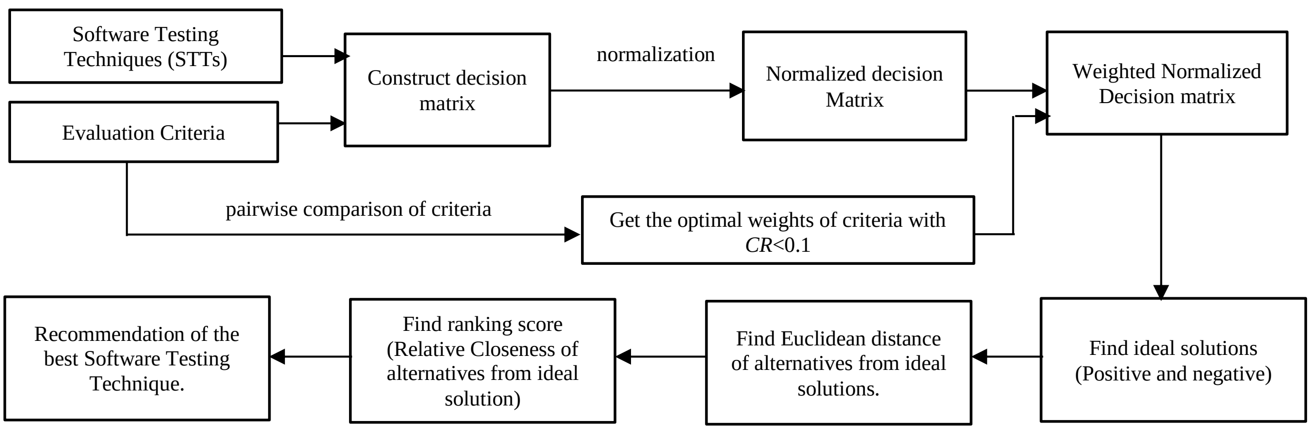
Software Testing Techniques (STTs)
Evaluation Criteria
Construct decision matrix
normalization
Normalized decision Matrix
Weighted Normalized Decision matrix
pairwise comparison of criteria
Get the optimal weights of criteria with CR<0.1
Find ideal solutions (Positive and negative)
Find Euclidean distance of alternatives from ideal solutions.
Find ranking score (Relative Closeness of alternatives from ideal solution)
Recommendation of the best Software Testing Technique.

Fig. 1. Graphical representation of the proposed methodology

Step3(d): Find the preference of all the criteria over the worst criterion using a number scale from 1 to 9. These preferences are represented in the form of others to worst vector as follows.

$$C_W = (c_{1W}, c_{2W}, \ldots.., c_{nW})^T \tag{4}$$

where c_{JW} denotes the preference of criteria j with respect to the worst criterion W. In this case $c_{WW} = 1$.

Step3(e): Find the optimal weights ($cw_1^*, cw_2^*, \ldots., cw_n^*$). Consider the weight of the criterion as an optimal weight where for each pair of cw_B / cw_j and cw_j / cw_W, we have $cw_B / cw_j = c_{BJ}$ and $cw_j / cw_W = c_{jW}$. For all values of j, these conditions may be satisfied by finding a solution where the maximum absolute differences $\left|\frac{cw_B}{cw_j} - c_{Bj}\right|$ and $\left|\frac{cw_j}{cw_W} - c_{jW}\right|$.

By taking into account the non-negativity and sum condition for the weights, the following problem is generated.

$$\min \max_j \left\{ \left|\frac{cw_B}{cw_j} - c_{Bj}\right|, \left|\frac{cw_j}{cw_W} - c_{jW}\right| \right\}$$
$$\text{such that}$$
$$\sum_j cw_j = 1 \text{ and } cw_j \geq 0, \text{ for all } j \tag{5}$$

The problem given in "(5)" can be converted into the following problem.

$$\begin{aligned} &\min \ \xi \\ &\text{such that} \\ &\left|\frac{cw_B}{cw_j} - c_{Bj}\right| \leq \xi, \ \left|\frac{cw_j}{cw_W} - c_{jW}\right| \leq \xi \text{ for all } j \\ &\sum_j cw_j = 1,\ cw_j \geq 0, \text{ for all } j \end{aligned} \tag{6}$$

The optimal weights ($cw_1^*, cw_2^*, \ldots., cw_n^*$) and ξ^* can be calculated by solving the problem given in "(6)". Here ξ^* is used to calculate the consistency ratio as described in the following step.

Step3(f): In this step, the Consistency Ratio (CR) is calculated using the following equation.

$$CR = \frac{\xi^*}{CI} \tag{7}$$

Here *CI* is the consistency index. *CI* is the maximum possible value of ξ for the different values of $c_{BW} \in \{1,2,3,.....,9\}$ listed in Table I. Comparisons will be more reliable if the value of *CR* is less. In general, if the value of *CR* is less than 0.1, comparisons are consistent.

TABLE I
CONSISTECY INDEX (*CI*) TABLE

c_{BW}	1	2	3	4	5	6	7	8	9
CI (max ξ)	0	0.44	1.00	1.63	2.30	3.00	3.73	4.47	5.23

Step4: Construction of Weighted Normalized Matrix $[WM]_{m\times n}$

Multiply each column of the normalized decision matrix [*NM*] by the respective column of the weight matrix [*W*] to obtain the weighted normalized matrix [*WM*]. The value of WM_{ij} of the weighted normalized matrix $[WM]_{m\times n}$ can be calculated by using the following equation

$$WM_{ij} = W_j \times NM_{ij} \tag{8}$$

Step5: Calculate Ideal Solutions $[PIS]_{n\times 1}$ and $[NIS]_{n\times 1}$

The best value each criterion may achieve is determined as the positive ideal solution. The least/worst value each criterion can achieve is used to calculate the negative ideal solution. They can be calculated as follows:

$$PIS = \begin{Bmatrix} (\max WM_{ij} \,/\, j \in z), \\ (\min WM_{ij} \,/\, j \in z') \text{ for i=1,2.....m} \end{Bmatrix} = \{WM_1^+, WM_2^+, WM_3^+, ...WM_n^+\} \tag{9}$$

$$NIS = \begin{Bmatrix} (\min WM_{ij} \,/\, j \in z), \\ (\max WM_{ij} \,/\, j \in z') \text{ for i=1,2.....m} \end{Bmatrix} = \{WM_1^-, WM_2^-, WM_3^-, ...WM_n^-\} \tag{10}$$

where *z* is associated with beneficial criteria and *z'* is associated with cost criteria.

Step6: Euclidean Distance

For each alternative Euclidean distance ED^+ from *PIS* and Euclidean distance ED^- from *NIS* is calculated using the following equations

$$ED_i^+ = \left\{ \sqrt{\sum_{j=1}^{n} (WM_{ij} - WM_j^+)^2} \text{ for i=1,2,.....,m} \right\} \tag{11}$$

$$ED_i^- = \left\{ \sqrt{\sum_{j=1}^{n} (WM_{ij} - WM_j^-)^2} \text{ for i=1,2,.....,m} \right\} \tag{12}$$

Step7: Find Relative Closeness $[RC]_{m\times 1}$

Relative closeness for each alternative with respect to negative ideal solution and positive ideal solution can be calculated using the following equation.

$$RC_i = \frac{ED_i^-}{(ED_i^- + ED_i^+)} \tag{13}$$

Step8: Selection of the best alternative

Rank the alternatives (in this study, software testing techniques) according to the value of relative closeness obtained in step 7. Software testing technique with the highest value of relative closeness (RC) will be recommended as most appropriate alternative.

IV. EXPERIMENTAL STUDY

A software was developed to maintain the research data of a university. It was necessary to test the integration between the unit-tested modules. Seven Software Testing Techniques (STT) and six Evaluation Criteria (EC) were identified, and the proposed methodology described in section III was applied to select the most suitable software testing technique. For testing the software, the following software testing techniques were identified.

STT= {Top-down integration testing, Bottom-up integration testing, Incremental integration, smoke testing, End-to-End testing, Big Bang, and Sandwich}.

The set of evaluation criteria considered for selecting the most appropriate software testing technique is as follows:

EC= {End user view, Test cases reusability, Fault detection time, Effort required for additional work, Test cases writing easiness, and Error fixing easiness}.

Stepwise application of the proposed method described in section III to select the best software testing technique is described as follows:

The decision matrix $[M]_{7\times 6}$ representing ratings of seven software testing techniques with respect to six evaluation criteria is shown in Table II.

TABLE II
DECISION MATRIX M

Software Testing Technique (STT)	Evaluation Criteria (EC)					
	EC1	*EC2*	*EC3*	*EC4*	*EC5*	*EC6*
STT1	3	4	5	3	4	3
STT2	2	1	2	3	4	1
STT3	1	3	3	5	4	5
STT4	5	3	4	3	4	4
STT5	4	5	4	4	5	3

STT6	4	2	5	3	3	1
STT7	2	4	5	3	3	4

Normalized decision matrix $[NM]_{7\times 6}$ can be calculated using "2" and is given in Table III.

TABLE III
NORMALIZED DECISION MATRIX $[NM]_{7\times 6}$

Software Testing Technique (STT)	Evaluation Criteria (EC)					
	EC1	*EC2*	*EC3*	*EC4*	*EC5*	*EC6*
STT1	0.3464	0.4472	0.4564	0.3235	0.3867	0.3419
STT2	0.2309	0.1118	0.1826	0.3235	0.3867	0.1140
STT3	0.1155	0.3354	0.2739	0.5392	0.3867	0.5698
STT4	0.5774	0.3354	0.3651	0.3235	0.3867	0.4558
STT5	0.4619	0.5590	0.3651	0.4313	0.4834	0.3419
STT6	0.4619	0.2236	0.4564	0.3235	0.2900	0.1140
STT7	0.2309	0.4472	0.4564	0.3235	0.2900	0.4558

Next, the weight matrix $[CW]_{1\times 6}$ can be constructed by using BWM [22] as described in the proposed methodology section (section III) and is shown in Table IV. The consistency ratio calculated is 0.035, which implies good consistency of judgments.

TABLE IV
WEIGHT MATRIX $[CW]_{1\times 6}$

Weights of Evaluation Criteria					
EC1	*EC2*	*EC3*	*EC4*	*EC5*	*EC6*
0.1787	0.1072	0.1341	0.0395	0.4064	0.1341

Next, the weighted normalized decision matrix $[WM]_{7\times 6}$ can be calculated using "8" and is shown in Table V.

TABLE V
WEIGHTED NORMALIZED DECISION MATRIX $[WM]_{7\times 6}$

Software Testing Technique (STT)	Evaluation Criteria (EC)					
	EC1	*EC2*	*EC3*	*EC4*	*EC5*	*EC6*
STT1	0.0619	0.0479	0.0612	0.0128	0.1572	0.0458
STT2	0.0413	0.0120	0.0245	0.0128	0.1572	0.0153
STT3	0.0206	0.0360	0.0367	0.0213	0.1572	0.0764
STT4	0.1032	0.0360	0.0490	0.0128	0.1572	0.0611
STT5	0.0825	0.0599	0.0490	0.0170	0.1964	0.0458
STT6	0.0825	0.0240	0.0612	0.0128	0.1179	0.0153
STT7	0.0413	0.0479	0.0612	0.0128	0.1179	0.0611

Use "9" to calculate the positive ideal (*PIS*) and use "10" to calculate the negative ideal solution (*NIS*). Calculated values of *PIS* and *NIS* are shown in Table VI.

TABLE VI
PIS AND *NIS*

Ideal Solution	Evaluation Criteria (EC)					
	EC1	*EC2*	*EC3*	*EC4*	*EC5*	*EC6*
PIS	0.1032	0.0599	0.0245	0.0128	0.1964	0.0764
NIS	0.0206	0.0120	0.0612	0.0213	0.1179	0.0153

Now, for each software testing technique, Euclidean distance ED^+ from *PIS* and Euclidean distance ED^- from *NIS* are calculated using "11" and "12". The ranking score of each software testing technique is calculated in terms of relative closeness using "13". Finally, ranks of software testing techniques are obtained, considering the higher the ranking score higher the rank will be. Euclidean distances, relative closeness, and ranks of software testing techniques are shown in Table VII.

TABLE VII
ED, ED, RC, AND RANKS OF SOFTWARE TESTING TECHNIQUES

Software Testing Techniques	ED^+	ED^-	RC	Rank
STT1	0.0753	0.0745	0.4972	3
STT2	0.1068	0.0582	0.3528	7
STT3	0.0957	0.0803	0.4564	4
STT4	0.0543	0.1061	0.6613	2
STT5	0.0445	0.1158	0.7225	1
STT6	0.1139	0.0636	0.3584	6
STT7	0.1083	0.0624	0.3655	5

V. RESULTS AND DISCUSSION

From Table VII, it can be observed that the relative closeness (RC) value for the End-to-End testing (STT5) is 0.7225 (highest), and hence End-to-End testing is declared the best software testing technique. End-to-end testing's properties, such as the exclusive focus on the end user's perspective and the reusability of test cases, which are some of the criteria used to evaluate testing approaches, make it the best option among the different available software testing techniques. This demonstrates that the methodology proposed considers all criteria for evaluating testing techniques and recommends the appropriate testing technique among available software testing techniques.

VI. Theoretical and Practical Implications

In present, testing techniques have gradually involved from the practice of single programmers or small development teams into a systematic, managed engineering discipline. Not only have there been numerous researches on testing techniques, but also more and more considerable industry practices. There are testing classes taught in universities. There have been special testing teams, test managers, and tester job positions open to professional testers; there have been training programs and complete procedures for testing in large enterprises; and there are increasing number of companies and vendors doing testing work for other companies. This study will be helpful to the future research scholars who want to do research in the field of software testing techniques.

The proposed method can be used in any software organization for selecting the most appropriate 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, and training needed 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 proposed method considers the subjective knowledge and practical aspects of the testing techniques in choosing the best testing technique to be used.

VII. Conclusion

Software testing plays a very important role in various software development phases. Selection of the right testing technique from different available testing techniques is critical as a testing technique can be chosen based on various factors. This paper proposes a hybrid approach BW-TOPSIS for selecting the right testing technique from different available software testing techniques by considering various factors (evaluation criteria). An experimental study was conducted to show the applicability and effectiveness of the proposed approach. Based on experimental results, it can be concluded that the proposed approach can be used to select the right testing technique at any stage of SDLC.

In this study we have used a smaller number of evaluation criteria. However, the proposed approach can be extended for the large number of evaluation criteria. Moreover, the proposed study does not consider the inter dependency between two criteria, considering the inter dependency of evaluation criteria may be another future research direction. The proposed hybrid approach can also be used in other decision-making problems in software engineering. For example, the proposed work can be extended to select the most appropriate SDLC model from different available SDLC models.

References

[1] I. Sommerville, Software Engineering, Addison-Wesley, 2011.

[2] E. Kazimieras Zavadskas, J. Antucheviciene, and P. Chatterjee, “Multiple-criteria decision-making (MCDM) techniques for business processes information management,” Information (Basel), vol. 10, no. 1, p. 4, 2018. https://doi.org/10.3390/info10010004

[3] T. E. J. Vos, B. Marin, M. J. Escalona, and A. Marchetto, “A methodological framework for evaluating software testing techniques and tools,” in 2012 12th International Conference on Quality Software, 2012. https://doi.org/10.1109/QSIC.2012.16

[4] K. Ajay Babu, K. Madhuri, and M. Suman, “An evaluation scheme of software testing strategy,” in Behavior Computing, London: Springer London, 2012, pp. 353–361. https://doi.org/10.1007/978-1-4471-2969-1_23

[5] O. A. L. Lemos, F. C. Ferrari, M. M. Eler, J. C. Maldonado, and P. C. Masiero, “Evaluation studies of software testing research in Brazil and in the world: A survey of two premier software engineering conferences,” J. Syst. Soft., vol. 86, no.4, pp. 951-969, 2013. http://dx.doi.org/10.1016/j.jss.2012.11.040

[6] A. C. Dias-Neto and G. H. Travassos, “Supporting the combined selection of model-based testing techniques,” IEEE trans. softw. eng., vol. 40, no. 10, pp. 1025–1041, 2014. https://doi.org/10.1109/TSE.2014.2312915

[7] M. Atifi, A. Mamouni, and A. Marzak, “A comparative study of software testing techniques,” in Networked Systems, Cham: Springer International Publishing, 2017, pp. 373–390. https://doi.org/10.1007/978-3-319-59647-1_27

[8] S. dalal, S. Hooda, and K. Solanki, “Comparative Analysis of Various Testing Techniques Used for Aspect-Oriented Software System ,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 12, no. 1, pp. 51-60, 2018. https://doi.org/10.11591/ijeecs.v12. i1.pp51-60

[9] R. Ibarra and G. Rodriguez, “SoTesTeR: Software testing techniques’ recommender system using a collaborative approach,” in Information Management and Big Data, Cham: Springer International Publishing, 2019, pp. 289–303. https://doi.org/10.1007/978-3-030-11680-4_28

[10] I. Shuaibu, M. Musa, and M. Ibrahim, “Investigation onto the software testing techniques and tools: An evaluation and comparative analysis,” Int. J. Comput. Appl., vol. 177, no. 23, pp. 24–30, 2019. https://doi.org/10.5120/ijca2019919685

[11] S. U. Farooq, “Gap between academia and industry: a case of empirical evaluation of three software testing methods,” Int. j. syst. assur. eng. manag., vol. 10, no. 6, pp. 1487-1504, 2019. https://doi.org/10.1007/s13198-019-00899-2.

[12] M. Khari and P. Kumar, “An extensive evaluation of search-based software testing: a review,” Soft Comput., vol. 23, no. 6, pp. 1933–1946, 2019. https://doi.org/10.1007/s00500-017-2906-y

[13] S. Sharma, L. Raja, and D. P. Bhatt, “Role of ontology in software testing,” J. Inf. Optimiz. Sci., vol. 41, no. 2, pp. 641–649, 2020. https://doi.org/10.1080/02522667.2020.1733196

[14] M. Qasim, A. Bibi, S. J. Hussain, N. Z. Jhanjhi, Mamoona Humayun, and N. U. Sama, “Test case prioritization techniques in software regression testing: An overview ,” International Journal of Advanced and Applied Sciences, vol. 8, no. 5, pp. 107-121, 2021. https://doi.org/10.21833/ijaas.2021.05.012

[15] T. Mårtensson, D. Ståhl, A. Martini, and J. Bosch, “Efficient and effective exploratory testing of large-scale software systems,” J. Syst. Softw., vol. 174, no. 110890, p. 110890, 2021. https://doi.org/10.1016/j.jss.2020.110890.

[16] S. Ali, Y. Hafeez, S. Hussain, and S. Yang, “Enhanced regression testing technique for agile software development and continuous integration strategies,” Softw. Qual. J., vol. 28, no. 2, pp. 397–423, 2020. https://doi.org/10.1007/s11219-019-09463-4

[17] K. Juhnke, M. Tichy, and F. Houdek, “Challenges concerning test case specifications in automotive software testing: assessment of frequency and criticality,” Softw. Qual. J., vol. 29, no. 1, pp. 39–100, 2021. https://doi.org/10.1007/s11219-020-09523-0

[18] S. W. G. AbuSalim, R. Ibrahim, and J. A. Wahab, “Comparative analysis of software testing techniques for mobile applications,” J. Phys. Conf. Ser., vol. 1793, no. 1, p. 012036, 2021. https://doi.org/10.1088/1742-6596/1793/1/012036

[19] D. Beyer, “First international competition on software testing,” Int. J. Softw. Tools Technol. Transf., vol. 23, no. 6, pp. 833–846, 2021. https://doi.org/10.1007/s10009-021-00613-3

[20] P. Jung, S. Kang, and J. Lee, "Reducing redundant test executions in software product line testing—A case study," Electronics (Basel), vol. 11, no. 7, p. 1165, 2022. https://doi.org/10.3390/electronics11071165

[21] C.L. Hwang, K. Yoon, Multiple Attribute Decision Making Methods and Applications, Springer, Berlin Heidelberg, 1981. https://doi.org/10.1007/978-3-642-48318-9.

[22] J. Rezaei, "Best-worst multi-criteria decision-making method," Omega, vol. 53, pp. 49–57, 2015. http://dx.doi.org/10.1016/j.omega.2014.11.009

Ajay Kumar is assistant professor with Department of Information Technology, KIET Group of Institutions, Delhi-NCR, Ghaziabad, India. He received his master degree in CSE from National Institute of Technical Teachers Training and Research, Chandigarh. He has done his Bachelor of Engineering degree in CSE from Dr. B. R. A University Agra, Uttar Pradesh, India. His research fields include software engineering, multi-criteria decision-making, and machine learning.

Kamaldeep Kaur is Associate Professor with University School of Information, Communication & Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She received her doctorate from Guru Gobind Singh Indraprastha University in 2016. Her research interests include Neural Networks, Natural Language Processing and Software Engineering. She is a lifetime member of Indian Society of Technical Education. She has published several research papers in Web of Science indexed journals and IEEE conferences. She is In-charge of IEEE Women in Engineering Chapter at University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University.