Computer Science, Engineering and Technology
Vol: 1(2), June 2023
REST Publisher; ISSN: 2583-9179 (Online)
Website: https://restpublisher.com/journals/cset/
DOI: https://doi.org/10.46632/cset/1/2/7

Developing A Testing Maturity Model for Software Test Process Evaluation and Improvement using the DEMATEL Method

***Elavarasi Kesavan**

Full Stack Quality Engineering Architect, Cognizant Mesa, AZ -85021, USA.

***Corresponding author Email ID:** elavarasikmk@gmail.com

Abstract: *Software process evaluation measures the effectiveness of the software processes employed in a software development organisation. The two prevalent evaluation techniques are SCE and ISO/IEC 15504. An endeavour to improve a software process can begin with a software process review. Software system assessment as well as enhancement go hand in hand with software process modelling. The most pertinent findings from both methods are reported in this work, and, With SCE and ISO/IEC 15504 being the two most popular evaluation methods, software process evaluation gauges the effectiveness of the software processes employed in software development organisations. The first step in any effort to improve a Software process evaluation and improvement are one thing, and software process modelling is another. In this study, the most pertinent findings from both methods are provided. Process improvement: Firms can identify areas for improvement in the method of developing software and put those improvements into practise to increase effectiveness, productivity, and quality. They can improve resource allocation, remove bottlenecks, and streamline procedures thanks to it. Quality assurance: Analysing the software manufacturing procedure enables the early detection of potential flaws and errors. Effective evaluation techniques can help organisations identify problems early and fix them before they have an impact on the final output. This enhances client happiness and helps deliver high-quality software to end users. Cost and resource management: Organisations can spot inefficiencies and wasteful resource allocation with the aid of a well-evaluated software development process. Organisations can lower development costs, increase resource utilisation, and more efficiently use budget and staff by optimising the process. The DEMATEL (Decision Making Trial and Evaluation Laboratory) method addresses a specific issue, pinup binding. Work through problems with a hierarchical structure. Contribute to identifying workable solutions. Structural modelling techniques are used for one reason: interrelationships between organizational components. Dependency identification and context It can affect the basic concept of relationships. and chart direction due to the influence of elements. makes more use of graphs. Requirements/ analysis, Design, Coding, Testing and Maintenance. The Rank using the DEMATEL for Software Development Process Evaluation in Requirements/ analysis is got the first rank whereas is the Design is having the Lowest rank.*
Keywords: *MCDM, Requirements/ analysis, Design, Coding, Testing and Maintenance.*

1. INTRODUCTION

Minimising the projected cost related to the purpose of software breakdown over the lifespan of a product is software testing and examination. To assess the efficacy of surveys and test scenarios, this article elaborates on finding defects events and malfunction detection incidents. connect the examinee's skills to the found problem. Similar to this, the procrastinating techniques employed to create the experimental settings are included in the impair trigger values [1]. By analysing the inspection and testing operations of some software products, it is possible to demonstrate the value of stimuli in assessing the efficacy of software inspections and tests. These analyses are used to identify weaknesses in both study and test strategies and to move towards strengthening them. In order to enhance the design, execution, and testing procedures, areas for additional research can be identified using the stimulus placement in an entire study or test series [2]. In current society, software systems are becoming more and more significant. Due to this, of creating software and the final product. In this article, process quality is the main topic. It is explained how a Test Maturity Model (TMM) was created to aid software development organisations in assessing and optimising their testing procedures. All actions pertaining to software quality are included in testing in its broadest definition [3]. We think that enhancing the testing procedure by fully utilising the TMM maturity criteria will significantly improve the calibre of software. Our study's objective may use to assess and enhance their testing procedure. To help achieve these goals, we recommend the following elements: a group of steps that specify the hierarchy of test maturity. Each level denotes a milestone in the development of a fully developed testing capacity. Moving up levels means that lower levels of behaviour are still being used [4]. Testing Process Maturity Definition A characterization of a competent test process is necessary before we can create a test maturity model. Work by Batek, Weber, and others. This gives the work a solid base. They contrasted and compared the behavioural traits of young and experienced software organisations [5]. They also go through the basic ideas that underlie the maturity of the software development process. We define assessment process maturation using their core ideas. Taking a cue from Falk, a regulated, measured, monitored, and efficient testing procedure [6]. Additionally, a sophisticated testing procedure is applied across the board, is backed by management, and is ingrained in the ethos of the company. Finally, a well-developed testing process offers the possibility for ongoing evolution and improvement. Constructing Objectives, tactics, test design specifications, and test case creation are all included in test planning. The test strategy should include outline the responsibility and resource distribution for testing the unit, communication system, and acceptability levels [7]. Evaluating a specific SDM's technical fitness for a project and their social suitability for a specific development team is the first step in improving the problem described above. The technical and social elements of SDMs have both been the subject of substantial research, but there is little overlap between the two areas of study, despite the fact that one of the goals of both is to increase the relevance of SDMs. Researchers frequently only consider one of two perspectives when examining SDMs, which leads to an incomplete assessment of SDMs. To obtain a comprehensive assessment, we think SDMs should be taken into account from both the technical and social perspectives [8]. The method assesses a research or testing A coding process involves activity and monitoring progression between distinct phases' activities as well as stages. The results of these analyses are delivered to the team that develops software and are used to determine the advantages and disadvantages of a research or testing activity. While specific measures aimed at improving the results of the present activity are taken after reported shortcomings, reported strengths signal the start of the subsequent activity [9]. By analysing the examination and validation operations of software products, this technique's value in assessing the effectiveness of software checks and evaluations is proved. These evaluations are intended to highlight flaws in research and testing procedures as well as the development of such tactics. An whole study or test series' stimulus distribution can be utilised to identify regions that need more research in order to enhance the design, execution, and testing procedures. This method assesses a research or test activity and monitors the progression through several phases and stages of the program's development process [10]. The results of these analyses are delivered to the team that develops software and are used to determine the advantages and disadvantages of a research or testing activity. Reporting strengths heralds the beginning of the upcoming action, whereas reported deficiencies are followed by specific measures designed to enhance the outcomes of the current activity. This method's value in evaluating the efficacy of software studies and tests can be determined by examining the study and testing efforts in software offerings is shown [11]. These evaluations are intended to highlight flaws in research and testing procedures as well as the development of such tactics. A whole study or test series' stimulus distribution can be utilised to identify regions that need more research in order to enhance the design, execution, and testing procedures. has a significant effect on the product's quality; the likelihood that a manufacturing phase will fail is strongly influenced by some earlier processes. This would suggest that the production process as a whole is dominated by uncertainty considerations. One is that many developing software efforts continue to fail, despite successful substantial research in the field [12]. Software of high calibre and dependability that complies with international norms and is simple to integrate into current system architectures is required. Additionally, the cost of developing and maintaining software is sharply rising, which leads to an increase in complexity and a demand for software that is better designed and easier to use. As a result, it is crucial to assess and evaluate these software properties [13]. The phrase "software evaluation" will be used to describe the evaluation of different software components throughout this essay. Selecting a single software product from a variety of software options to carry out a certain task is arguably the most frequent issue in software evaluation. In order to solve this issue, the focus of this research is on applications the evaluation framework for multi-criteria choice making (MCTM). Techniques for evaluating systems include rating, scoring, numerical optimisation, and making decisions based on many criteria [14]. Although the grading method is clear and intuitive, decision makers' (DMs') views are not accurately captured. The ranking system is constrained in the same way that the scoring system is. For resource optimisation for software selection, mathematical optimisation techniques such goal programming, 0-1 programming, and nonlinear optimisation were applied. However, complex mathematical models or limiting real-world implementation factors frequently limit the use of optimisation approaches. based on the multifaceted software performance characteristics [15]. To assist software development organisations in managing their processes successfully, a number of procedure development models as well as standards have been created. One such model is the systematic and rigorous techniques for process evaluation and improvement included in the capacity maturity model integration (CMMI). The International Organisation for Standardisation (ISO) created the rules and regulations that make up SPI. For instance, the efficacy of a business's computer systems is evaluated using ISO 9000 [16]. Software Process Excellence and Competency Determination (SPICE) uses ISO/IEC 15504 to improve processes. To evaluate and advance models as well as guidelines for improving processes, SPICE was created [17]. The more complex ISO/IEC 330XX family of processes evolved from the ISO/IEC 15504 standards. examination and enhancement standards, which encompass the evaluation of processes employed in an organisation, particularly maintaining them, management of changes, shipping, and refinement. These models and strategies can aid a company in producing a higher-quality product while spending fewer hours and dollars on it [18]. Process improvement initiatives have had some success, However, models and standards for enhancements to processes have not yet been fully created inside the context of GSD. It is crucial for procedure development workers to have a thorough grasp and expertise of SPI projects in an online setting because the majority of organisations are currently using GSD to reap numerous benefits [19]. The difficulties experienced by process teams working on projects in a GSD setting are very different from those in other contexts since SPI activities execution in a GSD environment is more complicated than collaborative development. Few studies have been done to create frameworks, models, and standards that can assist organisations in evaluating and implementing SPI activities in a GSD setting. To assist SPI practitioners in effectively measuring, evaluating, and improving their process improvement programmes, we suggest a model [20].

2. MATERIALS AND METHOD

Requirements/ analysis: Standards analysis, commonly referred to as requirements design, is the process of determining what users would expect from a new or modified product. It typically involves a team and calls for a variety of human soft skills, including critical thinking, communication, and judgement.
Design: The process of generating a specification for a software artefact that is subject to restrictions using a collection of simple components is known as software design.
Coding: Computers may follow rules created by coding. What a computer is able to do and cannot do is based on these instructions. Programmers can construct programmes, including apps and sites, by coding. Programming languages can instruct computers on how to analyse data more efficiently and quickly.
Testing: The practise of evaluating and verifying that a product or application that uses software operates as intended is known as testing its functionality. Testing has benefits including error prevention, decreased development costs, and increased productivity. Plan for managing tests.
Maintenance: Updating, modifying, and customising software to satisfy customer needs is known as software upkeep. Maintaining software is done after the item is released to enhance the overall software, correct issues or bugs, boost performance, and more.
Method: The DEMATEL method quickly separates the complex set of factors into a sender organization and a receiving institution, and then translates that information into the appropriate strategy for selecting a management tool. Also, the ZOGP model enables businesses to fully utilize their limited funds for planning to develop ideal management systems by combining different configurations with Explicit Priorities [21]. DEMATEL methods. This impact and causality can be attributed to affected group barricades. Therefore, to effectively implement electronic waste management, barriers belonging to a causally Influential subgroup should be given special consideration. Decision-makers must therefore identify hurdles in order to reduce their impact or influence, guarantee that the legal is strong, and ensure that appropriate barriers are in place [22]. Therefore, der methods ISM and DEMATEL methods, the results are somewhat consistent results grated ISM DEMATEL results for e-was determination constraints determine not only the structure of fire but also the structure of the interactions DEMATEL research, specific applications for DEMATEL. as for which DEMATEL is only. categories: factors or only relationships between criteria the first type of clarification is: and causal Group barriers pro or Source for affected group barriers can be considered due. Therefore, in order to effectively implement electronic waste management, barriers belonging to a causal or an influential group should be considered on a priority basis [23]. Therefore, decision makers need to determine obstacles the legal framework is strong make sure there is controllable in order to minimize impact or influence barriers. Therefore, derived structure of the interactions between these barriers is determined by the integrated ISM DEMATEL results for e-waste management constraints [24]. DEMATEL research, specific applications for DEMATEL. categories: factors or only relationships between criteria the first type of clarification involves identifying the main factors in terms of causal relationships and interrelationship size, while the second involves identifying the criteria for relationship and impact level analysis. DEMATEL method. As a result, the preliminary disadvantage (cluster one) was about topics such as the comparative weights of selection makers in the DEMATEL approach, which now does not take into account linking to team decision-making [25]. Obviously, in a group decision-making hassle, regular decision-makers can always trust their point of view and count on it to be prevalent among other selection-makers. This way, the very last evaluation guides must be close to their judgments, and if the very last assessment effects are close to their critiques, the choice maker is willing to simply accept it; otherwise, they may deny it. It is believed that methods based on unstructured comparisons, such as DEMATEL, play a significant role in the aforementioned discrepancies [26]. DEMATEL is widely accepted for analyzing the overall relationship of factors and classifying

factors into cause-and-effect types. Therefore, this article considers each source as a criterion in decision-making. To deal with a mixture of conflicting evidence, the significance and level of significance of each piece of evidence can be determined using DEMATEL; however, expanding the DEMATEL method with the source theory is required for better conclusions [27].

3. RESULT AND DISCUSSION

TABLE 1. Software Development Process Evaluation

	Requirements/ analysis	Design	Coding	Testing	Maintenance	Sum
Requirements/ analysis	0	19	17	15	18	69
Design	9	0	5	17	15	46
Coding	19	15	0	14	15	63
Testing	17	13	12	0	11	53
Maintenance	13	15	11	19	0	58

Table 1 shows that DEMATEL Decision making trail and evaluation laboratory in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance sum this value.

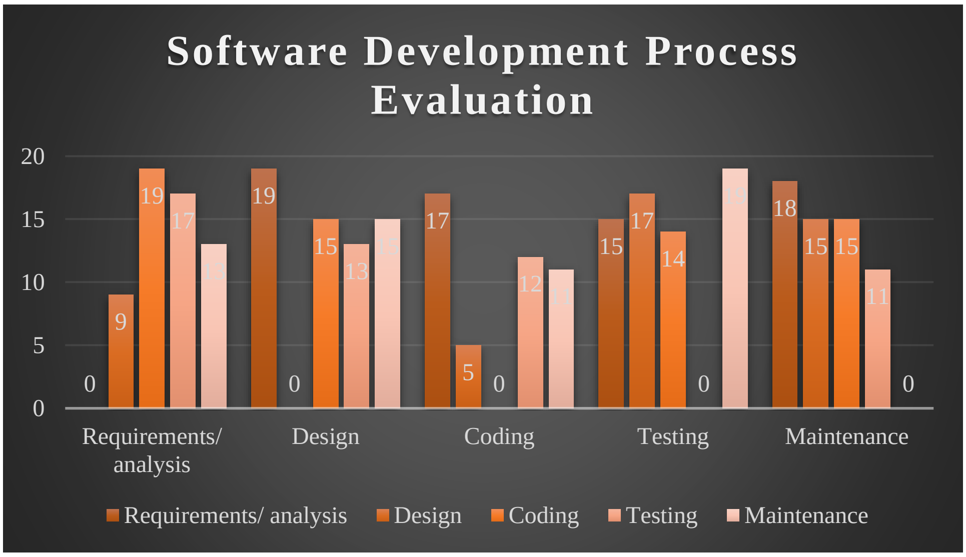

FIGURE 1. Software Development Process Evaluation

Figure 1 shows that DEMATEL Decision making trail and evaluation laboratory in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance sum this value.

TABLE 2. Normalization of Direct Relation Matrix

	Requirements/ analysis	Design	Coding	Testing	Maintenance
Requirements/ analysis	0	0.275362319	0.246376812	0.217391304	0.260869565
Design	0.130434783	0	0.072463768	0.246376812	0.217391304
Coding	0.275362319	0.217391304	0	0.202898551	0.217391304
Testing	0.246376812	0.188405797	0.173913043	0	0.15942029
Maintenance	0.188405797	0.217391304	0.15942029	0.275362319	0

Table 2 shows that the Normalizing of the direct relation matrix in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance the diagonal value of all the data set is zero.

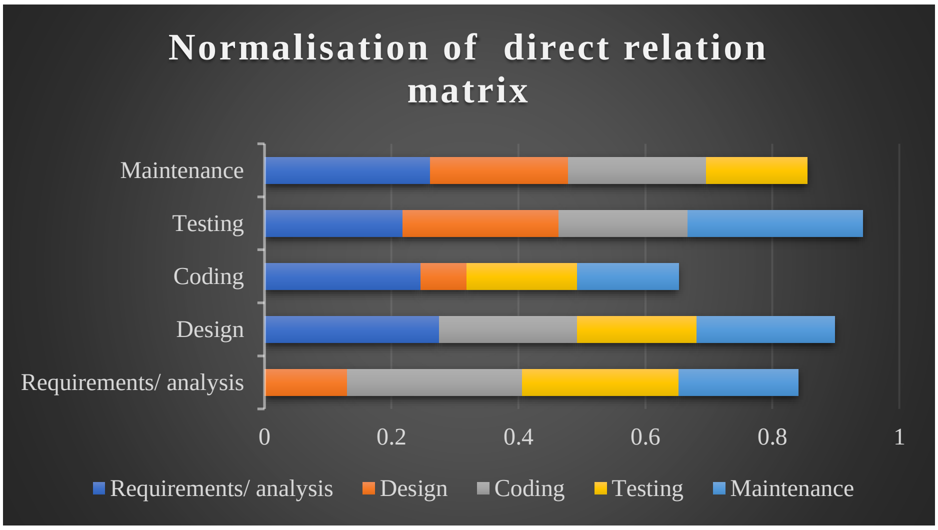

FIGURE 2. Normalization of Direct Relation Matrix

Figure 2 Shows that chart for Normalizing of direct relation matrix Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance the diagonal has Different value.

TABLE 3. Calculate the Total Relation Matrix

	Requirements/ analysis	**Design**	**Coding**	**Testing**	**Maintenance**
Requirements/ analysis	0	0.275362319	0.246376812	0.217391304	0.260869565
Design	0.130434783	0	0.072463768	0.246376812	0.217391304
Coding	0.275362319	0.217391304	0	0.202898551	0.217391304
Testing	0.246376812	0.188405797	0.173913043	0	0.15942029
Maintenance	0.188405797	0.217391304	0.15942029	0.275362319	0

Table 3 Shows the Calculate the total relation matrix in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance is Calculate the Value.

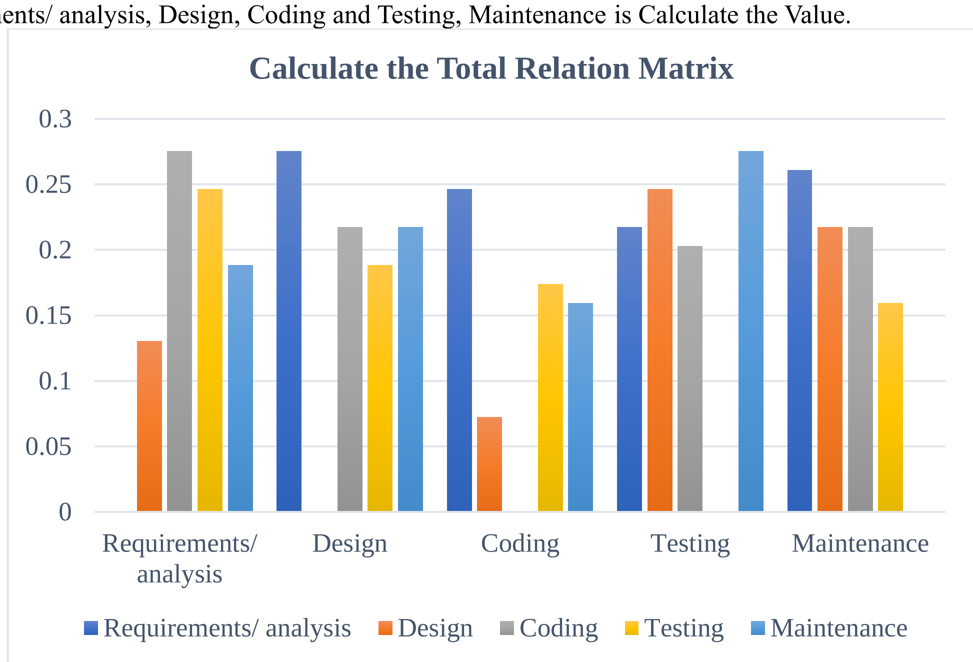

FIGURE 3. Calculate the Total Relation Matrix

Figure 3 Shows the Calculate the total relation matrix in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing, Maintenance is Calculate the Value.

TABLE 4. T= Y(I-Y)-1, I= Identity matrix

I				
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Table 4 Shows the T= Y(I-Y)-1, I= Identity matrix in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance is the common Value.

TABLE 5. Y Value

Y				
0	0.2753623	0.2463768	0.2173913	0.2608696
0.1304348	0	0.0724638	0.2463768	0.2173913
0.2753623	0.2173913	0	0.2028986	0.2173913
0.2463768	0.1884058	0.173913	0	0.1594203
0.1884058	0.2173913	0.1594203	0.2753623	0

Table 5 Shows the Y Value in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance is Calculate the total relation matrix Value and Y Value is the same value.

TABLE 6. I-Y Value

I-Y				
1	-0.275362319	-0.246376812	-0.217391304	-0.260869565
-0.130434783	1	-0.072463768	-0.246376812	-0.217391304
-0.275362319	-0.217391304	1	-0.202898551	-0.217391304
-0.246376812	-0.188405797	-0.173913043	1	-0.15942029
-0.188405797	-0.217391304	-0.15942029	-0.275362319	1

Table 6 Shows the I-Y Value in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance table 4 T= Y(I-Y)-1, I= Identity matrix and table 5 Y Value Subtraction Value.

TABLE 7. (I-Y)-1 Value

(I-Y)-1				
1.984256264	1.268368455	0.99706193	1.280519894	1.214257788
0.807115864	1.73529332	0.634510675	0.974347333	0.88105748
1.143771056	1.167613903	1.754906269	1.202271871	1.125372061
1.000412351	1.014964848	0.802150678	1.894085469	0.957957904
1.007121531	1.081830069	0.826439088	1.166298306	1.863500091

Table 7 shows the (I-Y)-1Value in Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance Table 6 shows the Minverse shows used.

TABLE 8. Total Relation matrix (T)

	Total Relation matrix (T)					Ri
Requirements/ analysis	0.890832	1.100689	1.168345	1.038156	1.010775	**5.208797**
Design	1.081081	0.837838	0.963964	0.864865	0.873874	**4.621622**
Coding	0.749868	0.735559	0.612259	0.81558	0.633104	**3.54637**
Testing	0.788553	0.952305	0.832538	0.666137	0.766826	**4.006359**
Maintenance	1.020138	1.195019	0.936584	1.031797	0.768239	**4.951775**
Ci	**4.530472**	**4.82141**	**4.51369**	**4.416534**	**4.052818**	

Table 8 shows the Total Relation Matrix (T) the direct relation matrix is multiplied by the inverse of the value that the direct relation matrix is subtracted from the identity matrix.

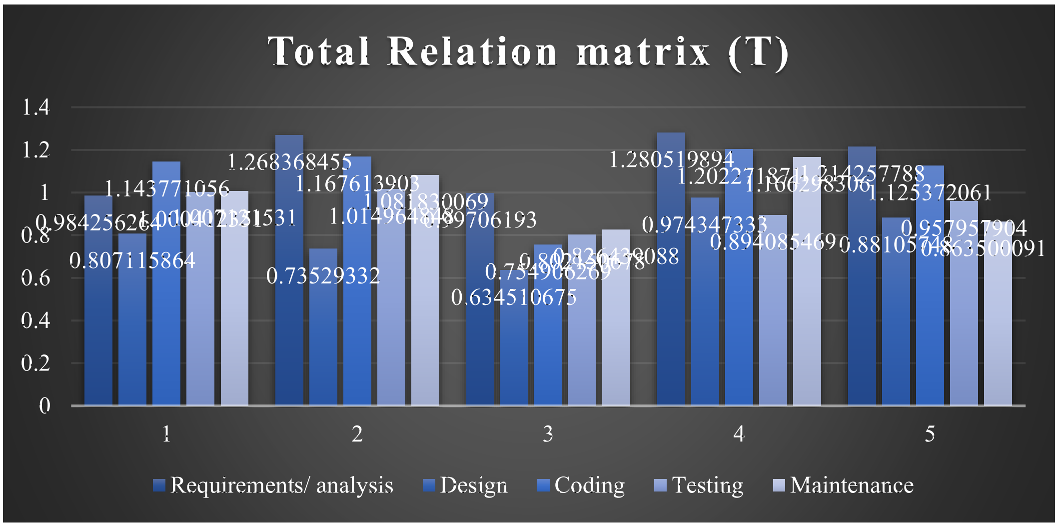

FIGURE 4. Total Relation matrix (T)

Figure 4. shows the Total Relation Matrix (T) the direct relation matrix is multiplied with the inverse of the value that the direct relation matrix is subtracted from the identity matrix.

TABLE 9. Software Development Process Evaluation Ri & Ci Value

	Ri	Ci
Requirements/ analysis	5.744464329	4.942677065
Design	4.03232467	5.268070594
Coding	5.39393516	4.015068639
Testing	4.669571249	5.517522872
Maintenance	4.945189084	5.042145323

Table 9 shows the Software Development Process Evaluation Ri, Ci Value Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance in Requirements/ analysis is showing the Highest Value for Ri and Design is showing the lowest value. Testing is showing the Highest Value for Ci and Coding is showing the lowest value.

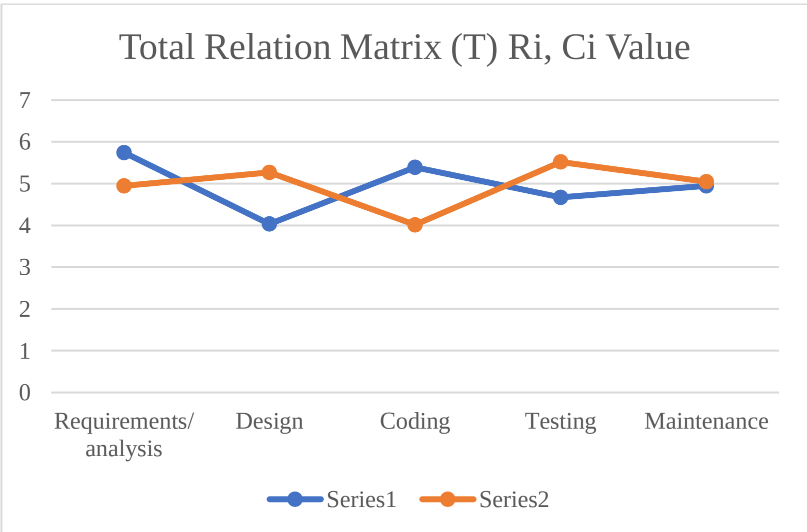

FIGURE 5. Total Relation Matrix (T) Ri, Ci Value

Figure 5 shows the Total Relation Matrix (T) Ri, Ci Value Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance in Requirements/ analysis is showing the Highest Value for Ri and Design is showing the lowest value. Testing is showing the Highest Value for Ci and Coding is showing the lowest value.

TABLE 10. Calculation of Ri+Ci and Ri-Ci to Get the Cause and Effect

	Ri+Ci	**Ri-Ci**	**Rank**	**Identity**
Requirements/ analysis	10.68714139	0.801787264	1	cause
Design	9.300395265	-1.235745924	5	effect
Coding	9.409003799	1.378866521	4	cause
Testing	10.18709412	-0.847951623	2	effect
Maintenance	9.987334407	-0.096956238	3	effect

Table 10 shows the Calculation of Ri+Ci and Ri-Ci to Get the Cause and Effect. Software Development Process Evaluation with respect to Requirements/ analysis, Design, Coding and Testing and Maintenance of Requirements/ analysis and Coding is Showing the highest Value of cause. Design, Testing and Maintenance is showing the lowest Value of effect.

TABLE 11. T matrix value

T matrix				
0.984256264	**1.268368**	**0.997062**	**1.28052**	**1.214258**
0.807115864	0.7352933	0.6345107	0.9743473	0.8810575
1.143771056	**1.167614**	0.7549063	**1.202272**	**1.125372**
1.000412351	**1.014965**	0.8021507	0.8940855	0.9579579
1.007121531	**1.08183**	0.8264391	**1.166298**	0.8635001

Table 11. Shows the T matrix calculate the average of the matrix and its threshold value (alpha) **Alpha 0.99141938** If the T matrix value is greater than threshold value then bold it

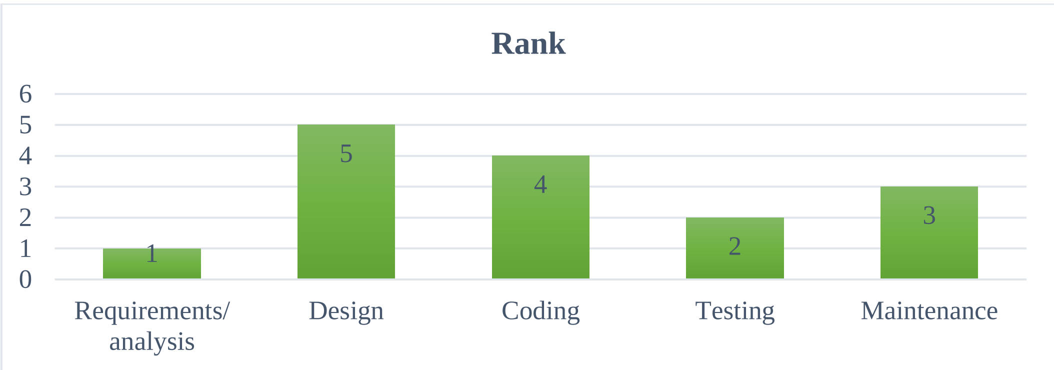

FIGURE 6. Shown the Rank

Figure 6 shows the Rank using the DEMATEL for Software Development Process Evaluation in Requirements/ analysis is got the first rank whereas is the Design is having the Lowest rank.

4. CONCLUSION

The first step in any effort to improve a Software process evaluation and improvement are one thing, and software process modelling is another. In this study, the most pertinent findings from both methods are provided. Process improvement: Firms can identify areas for improvement in the method of developing software and put those improvements into practise to increase effectiveness, productivity, and quality. They can improve resource allocation, remove bottlenecks, and streamline procedures thanks to it. Quality assurance: Analysing the software manufacturing procedure enables the early detection of potential flaws and errors. Effective evaluation techniques can help organisations identify problems early and fix them before they have an impact on the final output. This enhances client happiness and helps deliver high-quality software to end users. Cost and resource management: Organisations can spot inefficiencies and wasteful resource allocation with the aid of a well-evaluated software development process. Organisations can lower development costs, increase resource utilisation, and more efficiently use budget and staff by optimising the process. Minimising the projected cost related to The purpose of software breakdown over the lifespan of a product is software testing and examination. To assess the efficacy of surveys and test scenarios, this article elaborates on finding defects events and malfunction detection incidents. connect the examinee's skills to the found problem. Similar to this, the procrastinating techniques employed to create the experimental settings are included in the impair trigger values [1]. By analysing the inspection and testing operations of some software products, it is possible to demonstrate the value of stimuli in assessing the efficacy of software inspections and tests. Standards analysis, commonly referred to as requirements design, is the process of determining what users would expect from a new or modified product. It typically involves a team and calls for a variety of human soft skills, including critical thinking, communication, and judgement. The process of generating a specification for a software artefact that is subject to restrictions using a collection of simple components is known as software design. Computers may follow rules created by coding. What a computer is able to do and cannot do is based on these instructions. Programmers can construct programmes, including apps and sites, by coding. Programming languages can instruct computers on how to analyse data more efficiently and quickly. The Rank using the DEMATEL for Software Development Process Evaluation in Requirements/ analysis is got the first rank whereas is the Design is having the Lowest rank.

REFERENCES

[1]. Burnstein, Ilene, Taratip Suwanassart, and Robert Carlson. "Developing a testing maturity model for software test process evaluation and improvement." In Proceedings International Test Conference 1996. Test and Design Validity, pp. 581-589. IEEE, 1996.
[2]. Vavpotic, Damjan, and Marko Bajec. "An approach for concurrent evaluation of technical and social aspects of software development methodologies." Information and software technology 51, no. 2 (2009): 528-545.
[3]. Chaar, Jarir K., Michael J. Halliday, Inderpal S. Bhandari, and Ram Chillarege. "In-process evaluation for software inspection and test." IEEE transactions on Software Engineering 19, no. 11 (1993): 1055-1070.
[4]. Büyüközkan, Gülçin, and Da Ruan. "Evaluation of software development projects using a fuzzy multi-criteria decision approach." Mathematics and Computers in Simulation 77, no. 5-6 (2008): 464-475.
[5]. Khan, Arif Ali, Jacky W. Keung, and M. Abdullah-Al-Wadud. "SPIIMM: toward a model for software process improvement implementation and management in global software development." IEEE Access 5 (2017): 13720-13741.
[6]. Selby, Richard W., Victor R. Basili, and F. Terry Baker. "Cleanroom software development: An empirical evaluation." IEEE Transactions on Software Engineering 9 (1987): 1027-1037.
[7]. Xu, Peng, and Balasubramaniam Ramesh. "Using process tailoring to manage software development challenges." IT Professional 10, no. 4 (2008): 39-45.
[8]. Göransson, Bengt, Jan Gulliksen, and Inger Boivie. "The usability design process–integrating user-centered systems design in the software development process." Software Process: Improvement and Practice 8, no. 2 (2003): 111-131.
[9]. Bak, Jakob Otkjær, Kim Nguyen, Peter Risgaard, and Jan Stage. "Obstacles to usability evaluation in practice: a survey of software development organizations." In Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges, pp. 23-32. 2008.
[10]. Martin, Robert, and David Raffo. "Application of a hybrid process simulation model to a software development project." Journal of Systems and Software 59, no. 3 (2001): 237-246.
[11]. Weiss, David M., and Victor R. Basili. "Evaluating software development by analysis of changes: Some data from the software engineering laboratory." IEEE Transactions on Software Engineering 2 (1985): 157-168.
[12]. Wilkie, F. George, Donald McFall, and Fergal McCaffery. "An evaluation of CMMI process areas for small-to medium-sized software development organisations." Software Process: Improvement and Practice 10, no. 2 (2005): 189-201.
[13]. Procaccianti, Giuseppe, Héctor Fernández, and Patricia Lago. "Empirical evaluation of two best practices for energy-efficient software development." Journal of Systems and Software 117 (2016): 185-198.
[14]. Low, Graham C., and D. Ross Jeffery. "Function points in the estimation and evaluation of the software process." IEEE transactions on Software Engineering 16, no. 1 (1990): 64-71.
[15]. Shameem, Mohammad, Rakesh Ranjan Kumar, Chiranjeev Kumar, Bibhas Chandra, and Arif Ali Khan. "Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process." Journal of Software: Evolution and Process 30, no. 11 (2018): e1979.
[16]. Balsamo, Simonetta, Antinisca Di Marco, Paola Inverardi, and Marta Simeoni. "Model-based performance prediction in software development: A survey." IEEE Transactions on Software Engineering 30, no. 5 (2004): 295-310.
[17]. Islam, Shareeful, Haralambos Mouratidis, and Edgar R. Weippl. "An empirical study on the implementation and evaluation of a goal-driven software development risk management model." Information and Software Technology 56, no. 2 (2014): 117-133.
[18]. Smits, Hubert, and Guy Pshigoda. "Implementing scrum in a distributed software development organization." In Agile 2007 (AGILE 2007), pp. 371-375. IEEE, 2007.
[19]. Al-Zewairi, Malek, Mariam Biltawi, Wael Etaiwi, and Adnan Shaout. "Agile software development methodologies: Survey of surveys." Journal of Computer and Communications 5, no. 05 (2017): 74.

[20]. Case, Albert F. "Computer-aided software engineering (CASE) technology for improving software development productivity." ACM SIGMIS Database: the DATABASE for Advances in Information Systems 17, no. 1 (1985): 35-43.
[21]. Lee, Wen-Shiung, Alex YiHou Huang, Yong-Yang Chang, and Chiao-Ming Cheng. "Analysis of decision making factors for equity investment by DEMATEL and Analytic Network Process." Expert Systems with Applications 38, no. 7 (2011): 8375-8383.
[22]. Tsai, Wen-Hsien, and Wen-Chin Chou. "Selecting management systems for sustainable development in SMEs: A novel hybrid model based on DEMATEL, ANP, and ZOGP." Expert systems with applications 36, no. 2 (2009): 1444-1458.
[23]. Kumar, Ashwani, and Gaurav Dixit. "An analysis of barriers affecting the implementation of e-waste management practices in India: A novel ISM-DEMATEL approach." Sustainable Production and Consumption 14 (2018): 36-52.
[24]. Si, Sheng-Li, Xiao-Yue You, Hu-Chen Liu, and Ping Zhang. "DEMATEL technique: A systematic review of the state-of-the-art literature on methodologies and applications." Mathematical Problems in Engineering 2018 (2018).
[25]. Yazdi, Mohammad, Faisal Khan, RouzbehAbbassi, and RiszaRusli. "Improved DEMATEL methodology for effective safety management decision-making." Safety science 127 (2020): 104705.
[26]. Zhang, Weiquan, and Yong Deng. "Combining conflicting evidence using the DEMATEL method." Soft computing 23, no. 17 (2019): 8207-8216.
[27]. Lee, Hsuan-Shih, Gwo-HshiungTzeng, WeichungYeih, Yu-Jie Wang, and Shing-Chih Yang. "Revised DEMATEL: resolving the infeasibility of DEMATEL." Applied Mathematical Modelling 37, no. 10-11.