International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Design of Software Reliability Growth Model for Improving Accuracy in the Software Development Life Cycle (SDLC)

[1]Amol K. Kadam, [2]Dr. Konda Hari Krishna, [3]Neeraj Varshney, [4]A. Deepak, [5]Hemant Singh Pokhariya, [6]Dr. Sandeep Kumar Hegde, [7]Dr. Vinod H. Patil

Submitted: 24/06/2023 **Revised**: 06/08/2023 **Accepted**: 29/08/2023

Abstract: Software Testing is an essential activity primarily to check the quality of the software. Software testing is necessary for checking the gap between the expectations of the requirements stated by the client and the functionalities of the software after the implementation. Testing is becoming an important milestone in the process of developing software. Executing tests is a crucial phase of project development. The testing process for software uses a lot of testing resources, including tester, the quantity of test cases run, and processor time. Software quality is becoming more important in today's competitive market. Software testing is the process of identifying faults in all sophisticated application software that is put through several programming phases. Software testing helps to identify potential bugs and errors in the software being developed. Longer software testing does not mean more reliable software. Optimal code should also be closed to ensure high software quality. Due to its complex nature, it is difficult to remove all bugs in software. Also called error correction. Defect generation is defined as the occurrence of defects in software that cause future generations. Software reliability is the capacity to operate poorly in a particular context under specific circumstances. The goal of a software reliability optimization model is to quantify the factors that influence the software's dependability, most notably the quantity of residual defects, application failure percentage, and software reliability. The software reliability development model is designed to identify software errors and deficiencies in the process of software implementation. In the existing Software Reliability Development model, sometimes the testing method fails to remove defects and defects and does not find the value of the software. Exam assessment is the assessment of efforts and grades using various methods, tools, and techniques at the chosen exam level. A misguided testing effort usually results in insufficient testing, which will cause the software system to fail after it is deployed to the organization. The most important problem in software testing is evaluation, which is inevitable, but usually done in a hurry, and those responsible only wait for the simplest.

***Keywords** -software testing, software reliability testing coverage, test point analysis, function point analysis*

1. Introduction

Testing is a crucial activity to make sure code quality. Huge organizations will have many development groups with their product being a full test group. Team managers should be able to properly set up their schedules and associated resources and estimates for the needed execution effort will be an extra criterion for choice since effort could be restrictive in following. An honest execution effort estimation approach will profit each tester code comes [1],[2],[3]. There's an estimation model associated with expertise a primarily based approach for execution effort [4][5][6].

Software Reliability is the likelihood of failure-free functioning in a particular environment, throughout a particular time period, and under a particular set of circumstances.[7][8] Growth in Software Reliability Models are created to calculate software reliability metrics like the amount of unresolved bugs, the percentage of software that fails, and software reliability. [9][10].

Software testing is the process of finding flaws in all sophisticated computer programmes as they move through the stages of the software building cycle[11][12]. programme testing aids in finding any defects and mistakes in the programme that has been created. Longer software testing does not mean more secure software. Optimal code should also be closed to ensure high software quality. Due to its complex nature, it is difficult to remove all bugs in software. It is also called error correction [13][14][15]. Defect generation is defined as the occurrence of defects in software that cause future generations.

[1]*Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India*
[2]*Associate Professor, Department of CSE, School of Computing, Mohan Babu University,*
Tirupati, Andhra Pradesh
[3]*Department of Computer Engineering and Application, GLA University, Mathura*
[4]*Department of EC Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences,*
Saveetha University, Chennai, Tamilnadu.
deepakarun@saveetha.com
[5]*Assistant Professor, Department of Computer Science and Engineering, Graphic Era Deemed to be University,*
Dehradun, Uttarakhand
[6]*Associate Professor, Department of Computer Science and Engineering, NITTE (Deemed to be University),*
NMAM Institute of Technology, NITTE – 574110, Karnataka
[7]*Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, Maharashtra, India*
vhpatil@bvucoep.edu.in

Test evaluation consists of evaluating effort and value using several methods, tools, and techniques at the selected test level. Mistaking the testing effort usually leads to insufficient testing, which in turn will cause the software systems to fail after they are deployed to the organization [16][17][18]. Evaluation, the most important task in software testing, is an unavoidable task, but it is usually carried out in a hurry, and those responsible expect only the simplest [19][20][21].

Wherever the most faults can occur, inputs and program components are the focus of tests. Defects are the major objective of testing, and they are typically promptly discovered by a variety of testing disciplines. The relationships between effort, schedule, and quality must be in harmony [22][23][24]. It is generally acknowledged that calculating only one of each of these aspects without taking the others into account may result in inaccurate estimates [25][26][27]. Traditional estimating models are created using fixed inputs and fixed outputs along with linear or nonlinear multivariate analysis.

2. Problem Definition and Research Approach:

To overview the research literature to trace limitations of existing Software Reliability Growth Models. To make the testing phase more powerful, there is a need of capturing the cumulative impact of testing time, testing coverage & testing efforts in the testing phase the of software development life cycle [28]. The proposed system Includes the collective effect of testing time, testing coverage, and testing efforts [29]. In that Testing time, testing coverage and functional point analysis are the code and give the suggestions to how to improve the performance of the software, and Testing Efforts are mainly used to provide accuracy in the cost of that software [30][31][32].

So, our problem definition is "Analysis and Design of Software Reliability Growth Model concurrence with Software Development Life Cycle" [33

3. Methodology in Proposed Research Work:

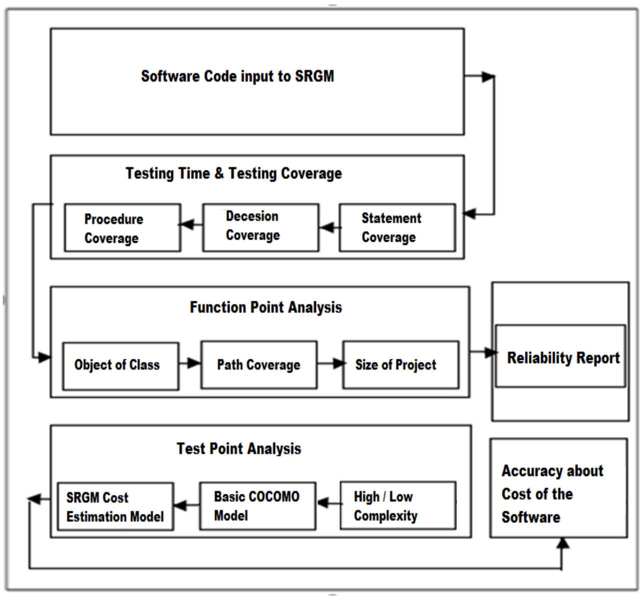


Fig. 1 Software Reliability Growth Model

Level 1 Input Software code to Software Reliability Growth Model:

Provides the input for the model in the form of the source code of the project. The code is stored in the form of a suitable data structure for further processing [34][35].

Level 2 Exam Time and Exam Period:

First, no line of code analysis, empty lines. Second, decision-making structures are identified and converted into numeric representations [36][37]. The cut-off for all classes has started. Finally, the procedure finds a weighted method with a functional description [38][39].

Level 3 - Analysis of the Function Points:

This level of deep code analysis is done to find the complexity of the software through functional analysis. In the function point analysis, the total effort is considered. Several objects were initialized with full code [40][41]. This value is compared to the recommended object limit. In addition, all class attributes are calculated [42][43]. Finally, the method functions written in the class is found, and these values are also calculated against the limit set. Finally, if the threshold is violated, the proposal is notified [44][45].

Level 4 -Analysis of the Test Points:

We focus mainly on the accuracy of software cost estimates. We bring that software complexity to the Basic COCOMO Model [46][47]. The complexity depends on the parameters such as Method, Boolean expression, object, line of code, and procedure environment [48][49].

3.1 Research Module:

3.1.1 Module-I

The Testing Time, Testing Environment, and [50] [51] Analysis of the Function Points:

In Module, I, the reliability of the project installed in the system is analysed to identify recommendations for improving reliability using threshold values [51][52][53].

Exam Time and Exam Environment:

Inhomogeneous Poison Process (EHPP):

• Test time: Also called Calendar time or central processing unit time [54].

• Test coverage: Predicting the reliability of the software is ensured using the testing environment [55].

Software developers use Test Coverage to assess the quality of software being tested and to help identify additional efforts needed to improve software reliability [56].

Cost of evacuation:

Default values are defined in functional testing and analysis environments. Refuge prices are quoted from researchers and industry experts[57][58][59].

Cost estimates depend on experience gained from the last project and historical data. When calculating the threshold value, the indicators considered in the sector are also considered[60][61][62]. Threshold values are monitored and updated through team experience and various processes[63][64].

Test size:

Coverage of the Statement: # of lines in the program. The threshold for the split is decided depending on the lines[65].

Coverage of the Conditions: Determines whether the Boolean expression tested in the control structure is true or false. If the Boolean expression is larger than the range limit, it is necessary to divide the Boolean expression[66][67].

• Procedure Coverage: Provides multiple procedures defined by Software Reliability Development Models (SRGM). It also advises the user if none of the procedures and functions exceeds the value set as threshold[68][69].

Function Point Analysis:

FPA i.e. function point analysis is a method of measuring the size of a computer application program using the complexity of the program point function[70].

• Count the number of objects in a class: The total number of objects in a class is determined as[71][72], Count the number of objects in a class for a given project for the system, and also count the number of attributes of that class. It is suggested to split the class if the total number of attributes in the class overshoots the limit.

• cover: Shows the number of valid paths available in tree.

• Estimate size of the project: In this step, Comments, lines of code, spaces etc counts and comments in the project and finally finds the total size of the application program.

The above methodology helps to find out the reliability and suggestions to improve the reliability of the software. When developers modify the software code using the recommendations provided by the system, and due to this the accuracy of the application increases.

Mathematical model of the test environment:

A non-uniform poison process for testing test coefficients:

The algebraic expression represents the quantitative stimulus from the mathematical model software analysis.

Block coverage: Block coverage is the total count of blocks processed by the test case.

Branch Branches: the total number of branches executed by the test case

$$\text{Block coverage} = \frac{\text{Test case covered by the No. of the blocks}}{\text{Total no of the blocks Available in Code}}$$

Test Case1:

$$\alpha_1 = \int_{t_0}^{t_1} c_1 \, dt$$

Where,

where

α1: The number of defects

t0 & t1: start time and end time

C1 is the cover function in the time interval t0 ≤ t < t1

Test Case 2:

$$\alpha_2 = \int_{t_1}^{t_2} c_2 \, dt$$

Where,

α_2 : No. of faults

C_2 is the coverage function

Resulting Test Case:

$$\alpha = \int_{t_0}^{t_1} c_1 \, dt + \int_{t_1}^{t_2} c_2 \, dt + \cdots + \int_{t_{n-1}}^{t_n} c_n \, dt$$

Where,

a is total no of faults

C_n is coverage function over interval time $t_{n-1} \leq t < t_n$

Advance ENHPP Model:

$$\alpha = \sum_{b=0}^{n} C(t_b)$$

Where α: Actual faults in software

3.1.2 Module-II

Test Point Analysis:

Determine the project's challenges using a number of characteristics, then communicate these challenges to COCOMO so that it can calculate the cost of this software programme.

TPA concentrate on the accuracy of software cost estimates. COCOMO provides software complexity for Models. Complexity mainly depends on five parameters including Method, Encapsulation, Object, Code Line, Environmental Procedure

A advance SRGM for cost estimation is: E = Complexity * ai(KLoC) (bi)

Where E represents the effort completed man-months,

KLoC indicates the number of 1000 lines of code executed, ci, bi, and ai are Constant values.

Table 3.1: COCOMO - Constant Values

Application Program	**a_i**	**b_i**	**c_i**	**d_i**
Embedded	3.6	1.20	2.5	0.32
Semi-Detached	3.00	1.120	2.51	0.350
Organic	2.4	1.05	2.5	0.38

The analysis of 30 projects has been performed to determine the high and low values. The KLoC of the project is shown in the table below

Difficulty: Small: 0.74, Big: 1.24.

Precision in pricing:

It is difficult to determine how many difficulties there are in the software because COCOMO does not identify the actual challenges; rather, it uses KLoC to help identify the business. However, the proposed model analyses the complexity and provides that complexity to COCOMO, which aids in achieving the business and accuracy during development and productivity.

Table 3.2: High and Low Range of Complexity

KLOC.	**Parameters**	**Complexities**	
		Low Values	**High Values**
1-60	Procedure Coverage	0-25	<25
	Objects	0-45	<45
	Code Lines	0-6000	<6000
	Methods	0-60	<60
	D Coverage	0-25	<25

51-300	Methods	60-130	<130
	Procedure Coverage	31-60	<60
	Class Objects	56-110	<110
	D Coverage	31-75	<75
	Code Lines	6000-61000	<61000
Above 300	Procedure Coverage	61-110	<110
	Code Lines	4 lacks	<4 lacks
	D Coverage	36-100	<100
	Objects	101-150	<150
	Methods	121-180	<180

The comparison between Basic COCOMO and the proposed model is as mentioned below:

The Low Complexity:

- **COCOMO Model (Basic): Z=b_i(KLoC) $^{(ai)}$**

Total KLOC= 5310/1000

= 5.310

Efforts: Z[i]= b[i]* (KLoC) $^{(ai)}$

= (2.4)* (5.310 $^{(1.05)}$)

= 13.85 Man required

Development is: D[i]=(c[i])*(Z[i] $^{(di)}$)

= 2.5*13.85$^{.38}$

= 6.78 Months

Productivity is: P[i]=KLoC/Z[i]

=5.310/13.85

= 0.383 Per month

- **Proposed Model: E= Complexity* a_i(KLoC) $^{(bi)}$**

KLOC= 5310/1000

= 5.310

Efforts are : E[i]= Complexity * a[i]* (KLoC) $^{(bi)}$

= 2.4* 5.310 $^{(1.05)}$ * 0.74

= 10.24 Man-Month

Development is: D[i]= (c[i])* (E[i] $^{(di)}$)

= 2.5*10.24$^{.38}$

= 06.150 months

Productivity is: P[i]=(KLoC/E[i])

=(5.310/10.24)

= 0.521 Per month

The Large Difficulties:

- **COCOMO Model is : Z=a_i(KLoC) $^{(bi)}$**

KLOC= 15420/1000

= 15.420

Efforts are: (E[i])= (a[i]* (KLoC) $^{(bi)}$)

= 2.4* 15.420 $^{(1.05)}$

= 42.43 Man per Month

Development is : D[I]=c[i]* E[i] $^{(di)}$

= 2.5*42.43$^{0.38}$

= 10.38 Months

Productivity is: P[i]=KLoC/E[i]

=15.420/42.43

= 0.363 Per month

- **Advance Model is : E= Complexity* a_i(KLoC) $^{(bi)}$**

KLOC= 15420/1000

= 15.420

Efforts are : E[i]= Complexity* a[i]* (KLoC) $^{(bi)}$

=1.24* 2.4* 15.420 $^{(1.05)}$

= 52.61 Man-Month

Development is : D[i]= (E[i]) $^{(di)}$ * (c[i])

= $52.61^{0.38}$ *2.5

= 11.25 Months

Productivity: P[i]= (KLoC)/(E[i])

=(15.420/52.61)

= 0.293 Per month

4. Results & Discussion:

Screen: Testing Time & Testing Coverage

Description: This screen contains four tabs for analysis performed for the software lines. The first tab gives input for the process. The browse tab is used to select a specific system folder as shown in the following figure.

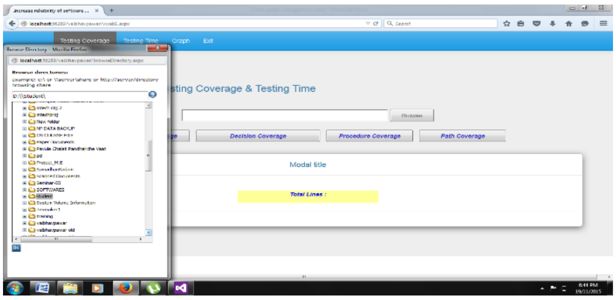


Fig 2: Browse the Software Project in our Model

Screen: Decision Screen

Description: This screen shows the Counts for the quantity of Boolean expressions executed in the project.

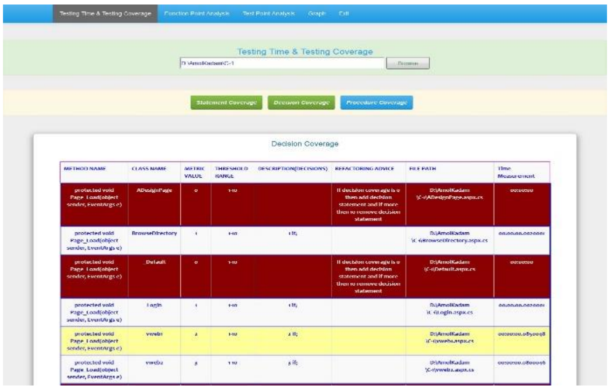


Fig.3: Find out the Decision Coverage

Display: Procedure screen

Description: This screen shows Count of the number of methods for each class in the project

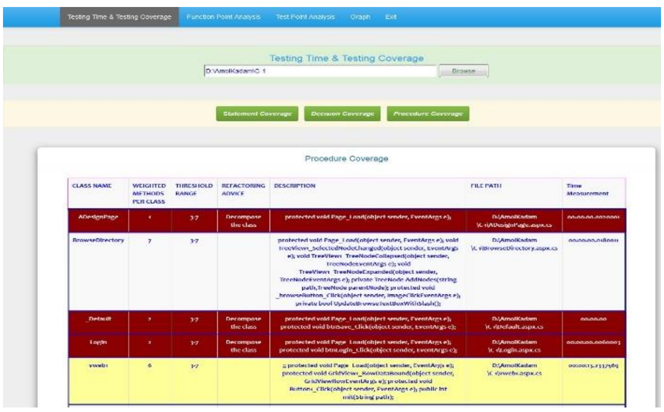


Fig 4: Find out the Procedure Coverage

Display: Count the number of class objects.

Description: On this screen, a count of several objects of every class of the software is displayed. If greater than 10 objects are present in the project, then it gives suggestions to the user.

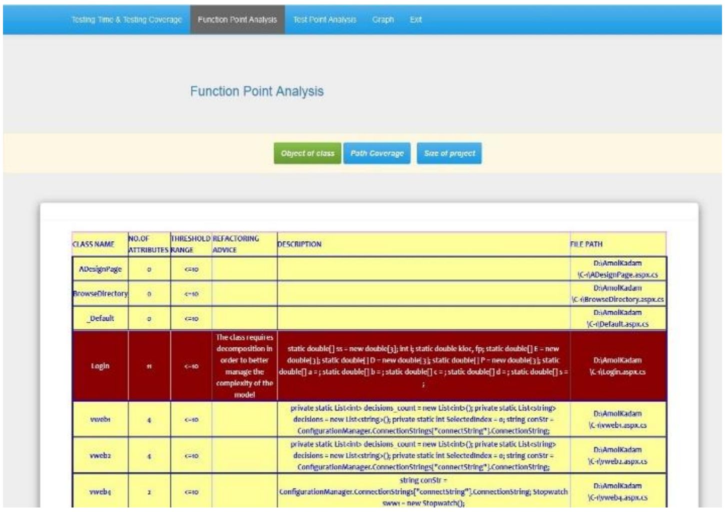


Fig 5: Find out the Object per Class

Screening: Calculate the coverage of the path

Description: When we click the button of the option path to view the path, it gives the file direction, class, values of metric metric, range, class name.

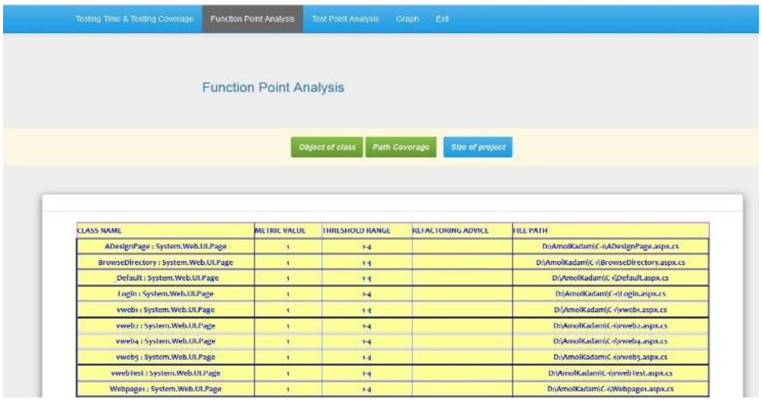


Fig 6: Find out Path Coverage

Screen: Determine the project size

description: Finally, it helps to know the total scope of the project

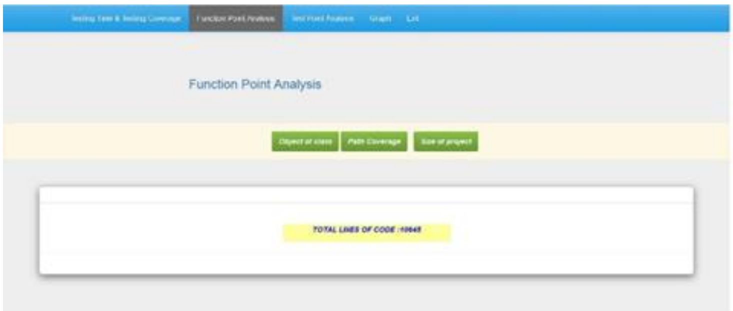
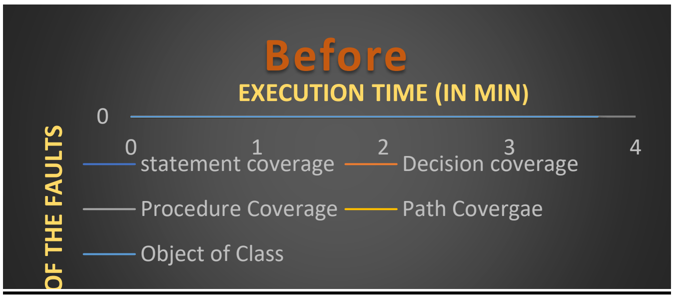


Fig 7: Find out the total size of the project.

5. Resultant Graphs:



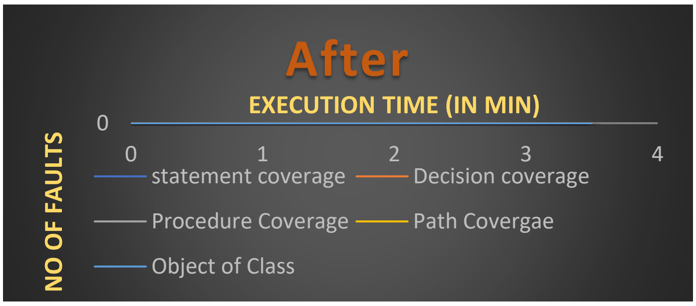


Fig 8: Before and after using Threshold values

The above graph represents the result of the analysis of the project. The bias in the results was greater, before using the threshold value indicating a higher software complexity.

After using the threshold value and updating the project based on the recommendations given, it determines that the deviation in the results is not compared to before using the threshold value, so when the developer uses this system for his project, it shows that the reliability of the software is higher than other systems exist.

5.1 Analysis of test point analysis results:

Comparison in three orders:

1) Embedded: Developed under strict constraints. A combination of organic and semi-structured projects.
2) Organic arrangement: Small group with good experience
3) Semi-Detached: Medium group with mixed experience

Table 4.1: The Big Complexity Project comparative

COCOMO				**Proposed**		
	KLoC= 7.729			**KLoC= 7.729**		
	Time	**Productivity**	**Effort**	**Time**	**Productivity**	**Effort**
Semi-detached	**8.19**	**0.26**	**29.72**	**8.86**	**0.2**	**37.15**
Organic Mode	**7.89**	**0.37**	**20.26**	**8.59**	**0.3**	**25.75**

Table 4.2: The Small Complexity Project comparative

COCOMO				**Proposed**		
	KLoC=3.327			**KLoC=3.327**		
	Time	**Productivity**	**Effort**	**Time**	**Productivity**	**Effort**
Organic Mode	5.63	0.39	8.47	5.04	0.52	6.35
Semi-detached	5.38	0.28	11.52	5.31	0.38	8.64

In the table above, a low-complexity project has been compared with a high-complexity project with low effort and high effort, COCOMO, but only needs to provide accuracy to determine whether the increase or decrease in the cost of the project depends on the complexity of the project.

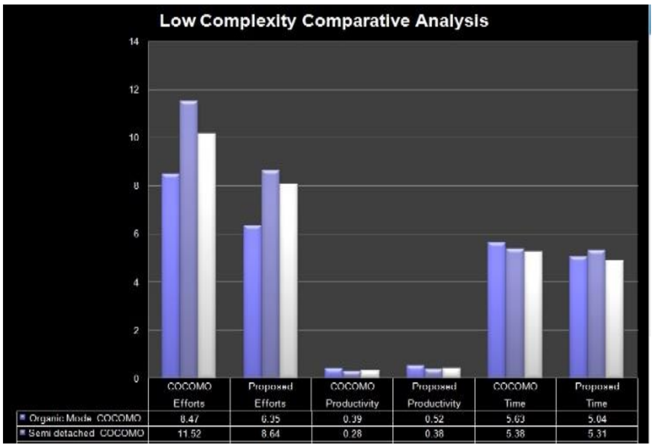


Fig 9: Comparison Graph for Small Complexity

We used COCOMO to analyse the outcomes of the less complexity project because COCOMO required less work and time, and it enhance productivity.

We analyzed the results with COCOMO in projects of high complexity because effort and time increased and the performance was lower than COCOMO because the software was very complex, even so confirmed the accuracy of the software cost estimate.

6. Conclusion:

The researcher was inadequate to develop a advanced software reliability growth model (SRGM), containing several framework such as test time, test environment, operational point analysis, and check point analysis. Whereas the execution of the Reliability Development Model assist to define reliability and provides recommendations on how to improve software reliability. SRGM can calculate project costs that cannot be calculated in a reliable software development model.

Academic research activities along with development activities can be studied with the help of SRGM design. By considering the price of refugees, the indicators used for the development of the sector are taken into account. Group rates are monitored, and they are updated based on the group's experience and the various activities used.

Growth’s proposed software reliability model is also useful in interdisciplinary research projects and consulting work.

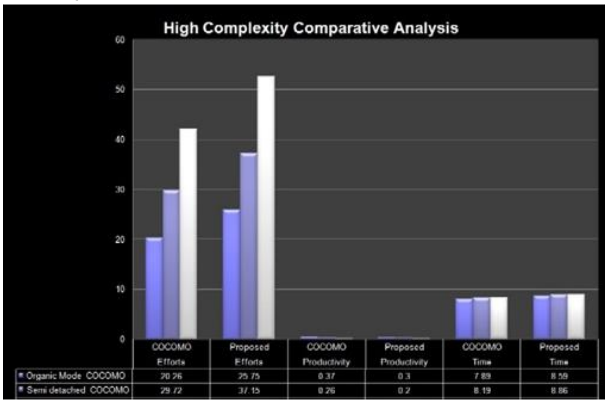


Fig 10: Comparison Graph for High Complexity

References

[1] Beldar, Kavita K., M. D. Gayakwad, and M. K. Beldar. 2016. “Optimizing Analytical Queries on Probabilistic Databases with Unmerged Duplicates Using MapReduce.” Int. J. Innov. Res. Comput. Commun. Eng 4: 9651–59.

[2] Beldar, Kavita K., M. D. Gayakwad, Debnath Bhattacharyya, and Hye-Jin Kim. 2016a. “Query Evaluation on Probabilistic Databases Using Indexing and MapReduce.” International Journal of Database Theory and Application 9 (10): 363–78.

[3] Beldar, Kavita K., M. D. Gayakwad, Debnath Bhattacharyya, and Tai-Hoon Kim. 2016b. “A Comparative Analysis on Contingence Structured Data Methodologies.” International Journal of Software Engineering and Its Applications 10 (5): 13–22.

[4] Beldar, Miss Menka K., M. D. Gayakwad, and Miss Kavita K. Beldar. 2018. “Altruistic Content Voting

System Using Crowdsourcing." International Journal of Scientific Research and Review 7 (5): 477–86.

[5] Beldar, Miss Menka K., M. D. Gayakwad, Miss Kavita K. Beldar, and M. K. Beldar. 2018. "Survey on Classification of Online Reviews Based on Social Networking." IJFRCSCE 4 (3): 55.

[6] Boukhari, Mahamat Adam, Prof Milnid Gayakwad, and Prof Dr Suhas Patil. 2019. "Survey on Inappropriate Content Detection in Online Social Media." International Journal of Innovative Research in Science, Engineering, and Technology 8 (9): 9297–9302.

[7] Gayakwad, M. D., and B. D. Phulpagar. 2013. "Research Article Review on Various Searching Methodologies and Comparative Analysis for Re-Ranking the Searched Results." International Journal of Recent Scientific Research 4: 1817–20.

[8] Gayakwad, Milind. 2011. "VLAN Implementation Using Ip over ATM." Journal of Engineering Research and Studies 2 (4): 186–92.

[9] Gayakwad, Milind, and Suhas Patil. 2020. "Content Modelling for Unbiased Information Analysis." Libr. Philos. Pract, 1–17.

[10] Gayakwad, Milind, Suhas Patil. "Analysis of Methodologies to Model the Content for Conveying the Correct Information." In 2021 International Conference on Computing, Communication and Green Engineering (CCGE), 1–4. IEEE.

[11] Gayakwad, Milind, Suhas Patil ."Assessment of Source, Medium, and Intercommunication for Assessing the Credibility of Content." In 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GenCon), 1–5. IEEE.

[12] Gayakwad, Milind, Suhas Patil, Rahul Joshi, Sudhanshu Gonge, and Sandeep Dwarkanath Pande. "Credibility Evaluation of User-Generated Content Using Novel Multinomial Classification Technique." International Journal on Recent and Innovation Trends in Computing and Communication 10 (2s): 151–57.

[13] Gayakwad, Milind, Suhas Patil, Amol Kadam, Shashank Joshi, Ketan Kotecha, Rahul Joshi, Sharnil Pandya, et al. 2022. "Credibility Analysis of User-Designed Content Using Machine Learning Techniques." Applied System Innovation 5 (2): 43.

[14] Harane, Swati T., Gajanan Bhole, and Milind Gayakwad. 2017. "SECURE SEARCH OVER ENCRYPTED DATA TECHNIQUES: SURVEY." International Journal of Advanced Research in Computer Science 8 (7).

[15] Kavita Shevale, Gajanan Bhole, Milind Gayakwad. 2017. "Literature Review on Probabilistic Threshold Query on Uncertain Data." International Journal of Current Research and Review 9 (6): 52482–84.

[16] Mahamat Adam Boukhari, Milind Gayakwad. 2019. "An Experimental Technique on Fake News Detection in Online Social Media." International Journal of Innovative Technology and Exploring Engineering (IJITEE) 8 (8S3): 526–30.

[17] Maurya, Maruti, and Milind Gayakwad. 2020. "People, Technologies, and Organizations Interactions in a Social Commerce Era." In Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2018), 836–49. Springer International Publishing.

[18] Milind Gayakwad, B. D. Phulpagar. 2013. "Requirement Specific Search." IJARCSSE 3 (11): 121.

[19] Panicker, Aishwarya, Milind Gayakwad, Sandeep Vanjale, Pramod Jadhav, Prakash Devale, and Suhas Patil. n.d. "Fake News Detection Using Machine Learning Framework."

[20] Sharma, Jitin, Prashant C. Chavan, T. B. Patil, Supriya C. Sawant, and Milind Gaykawad. 2022. "A Comparative Analysis of Brain Tumor Classification and Prediction Techniques by Applying MRI Images Encompassing SVM and CNN with Transfer Learning Method." Journal of Algebraic Statistics 13 (3): 393–405.

[21] Shevale, Kavita, Gajanan Bhole, and Milind Gayakwad. 2017. "Probabilistic Threshold Query on Uncertain Data Using SVM." Int. J. Adv. Res. Comput. Sci 8: 1967–69.

[22] Singh, Mahendra Kumar, Amol K. Kadam, Milind Gayakwad, Pramod Jadhav, Vinayak N. Patil, Prasad Kadam, Vinod Patil, and Sunita Dhotre. n.d. "An empirical approach for underwater image quality enhancement and object detection using deep learning."

[23] Yamada, Shigeru, Mitsuru Ohba, and Shunji Osaki. "S-shaped reliability growth modeling for software error detection." IEEE Transactions on Reliability 32.5 (1983): 475-484.

[24] Malaiya, Y. K., Li, M. N., Bieman, J. M., & Karcich, R. (2002). Software reliability growth with test coverage. IEEE Transactions on Reliability, 51(4), 420-426.

[25] Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. European Journal of Operational Research, 145(2), 443-454.

[26] Ledoux, J. (2003). Software reliability modeling. Handbook of Reliability Engineering, 213-234.

[27] Zheng, J. (2009). Predicting software reliability with neural network ensembles. Expert systems with applications, 36(2), 2116-2122.

[28] Martens, A., Koziolek, H., Becker, S., &Reussner, R. (2010, January). Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In Proceedings of

the first joint WOSP/SIPEW international conference on Performance engineering (pp. 105-116). ACM.

[29] Martens, A., Koziolek, H., Becker, S., & Reussner, R. (2010, January). Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering (pp. 105-116).ACM.

[30] Zio, E. (2009). Reliability engineering: Old problems and new challenges. Reliability Engineering & System Safety, 94(2), 125-141.

[31] Aljahdali, S., &Sheta, A. F. (2011, April). Predicting the reliability of software systems using fuzzy logic. In Information Technology: New Generations (ITNG), 2011 Eighth International Conference on (pp. 36-40). IEEE.

[32] Lohmor, S., &Sagar, B. B. (2014). Overview: Software Reliability Growth Models. Int. J. Comput. Sci. Inf. Technol.

[33] "Two Dimensional Flexible Software Reliability Growth Model with Two Types of Imperfect Debugging" P.K. Kapur1, Anu G. Aggarwal2 and Abhishek Tandon3 Department of Operational Research, University of Delhi,

[34] "A Novel Framework of Software Reliability Evaluation with Software Reliability Growth Models and Software Metrics" 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering.

[35] Jyoti G. Borade, Vikas R. Khalkar, "Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 8, August-2013, ISSN: 2277 128X.

[36] Lance Fiondella, Swapna S. Gokhale, "Optimal Allocation of Testing Effort Considering Software Architecture", IEEE Transactions on Reliability, June 2012 vol. 61, no. 2.

[37] Kamala RamasubramaniJayakumar, Alain Abran, "A Survey of Software Test Estimation Techniques", Journal of Software Engineering and Applications, Octomber-2013, 6, 47-52.

[38] Yuejian Wu, Paul MacDonald, *"*Testing ASICs with Multiple Identical Cores*",* IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, MARCH 2003, VOL. 22, NO. 3.

[39] AkitoMonden, Takuma Hayashi, Shoji Shinoda, Kumiko Shirai, Junichi Yoshida, Mike Barker, Kenichi Matsumoto, "Assessing the Cost Effectiveness of Fault Prediction in Acceptance Testing", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, OCTOBER 2013, VOL. 39, NO. 10.

[40] Chin-Yu Huang, Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function Into Software Reliability Modeling", IEEE TRANSACTIONS ON RELIABILITY, SEPTEMBER 2002, VOL. 51, NO. 3.

[41] Yashwant K. Malaiya, MichaelNaixin Li, James M. Bieman, Rick Karcich, "Software Reliability Growth With Test Coverage", IEEE TRANSACTIONS ON RELIABILITY, DECEMBER 2002 VOL. 51, NO. 4.

[42] Sy-Yen Kuo, Chin-Yu Huang, Michael R. Lyu, "Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates", IEEE TRANSACTIONS ON RELIABILITY, SEPTEMBER 2001, VOL. 50, NO. 3.

[43] Dr. N. Balaji, N. Shivakumar & V. Vignaraj Ananth," Software Cost Estimation using Function Point with Non-Algorithmic Approach", Global Journal of Computer Science and Technology Software & Data Engineering Volume 13 Issue 8 Version 1.0 the Year 2013

[44] Pawar, V. E., Kadam, A. K., & Joshi, S. D. (2015). Analysis of Software Reliability using Testing Time and Testing Coverage. International Journal of Advance Research in Computer Science and Management Studies.

[45] Washizaki, H., Honda, K., & Fukazawa, Y. (2015, August). Predicting release time for open source software based on the generalized software reliability model. In Agile Conference (AGILE), 2015 (pp. 76-81). IEEE.

[46] Pawar, V. E., Kadam, A. K., & Joshi, S. D. (2015). Analysis of Software Reliability using Testing Time and Testing Coverage. International Journal of Advance Research in Computer Science and Management Studies.

[47] Mane, M., Joshi, M., Kadam, A., & Joshi, S. D. (2014). Software Reliability and Quality Analyzer with Quality Metric Analysis Along With Software Reliability Growth Model. International Journal of Computer Science & Information Technologies, 5(3).

[48] Sabnis, P., & Kadam, A. Software Reliability Growth Model with Bug Cycle and Duplicate Detection Techniques.

[49] Kadam, A. K., Joshi, S. D., Bhattacharyya, D., & Kim, H. J. (2016). Software Superiority Achievement through Functional Point and Test Point Analysis. International Journal of Software Engineering and Its Applications, 10(11), 181-192.

[50] Iqbal, J. (2017). Software reliability growth models: A comparison of linear and exponential fault content functions for study of imperfect debugging situations. Cogent Engineering, 4(1), 1286739.

[51] Choudhary, A., Baghel, A. S., & Sangwan, O. P. (2016, January). Software reliability prediction modeling: a comparison of parametric and non-parametric modeling. In Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference (pp. 649-653). IEEE.

[52] Kumar, A. (2016, February). Software Reliability Growth Models, Tools, and Data Sets-A Review. In Proceedings of the 9th India Software Engineering Conference (pp. 80-88).ACM.

[53] Mao, C., & Li, Q. (2016, August). A Testing-Coverage Software Reliability Growth Model Considering the Randomness of the Field Environment. In Software Quality, Reliability and Security Companion (QRS-C), 2016 IEEE International Conference on (pp. 402-403). IEEE.

[54] Zhu, M., & Pham, H. (2016). A software reliability model with time-dependent fault detection and fault removal. Vietnam Journal of Computer Science, 3(2), 71-79.

[55] Nagaraju, V., &Fiondella, L. (2017, January). A single change point software reliability growth model with heterogeneous fault detection processes. In Reliability and Maintainability Symposium (RAMS), 2017 Annual (pp. 1-6). IEEE.

[56] Chi, J., Honda, K., Washizaki, H., Fukazawa, Y., Munakata, K., Morita, S., & Yamamoto, R. (2017, March). Defect Analysis and Prediction by Applying the Multistage Software Reliability Growth Model. In Empirical Software Engineering in Practice (IWESEP), 2017 8th International Workshop on (pp. 7-11). IEEE.

[57] Lohmor, S., & Sagar, B. B. (2014). Overview: Software Reliability Growth Models. Int. J. Comput. Sci. Inf. Technol.

[58] Ashwini Kurhade, J. Naveenkumar, A. K. Kadam, “Efficient Algorithm TKO with TKU for Mining Top-K Item Set,” vol. 11, no. 05, pp.1566-1570, 2019.

[59] P. K. Suryawanshi, P. A. K. Kadam, P. S. S. Dhotre, and P. P. A. Jadhav, “A Novel Approach for Women Security with Information Fusion for Multi-Sensory Data,” vol. 8, no. 1, pp. 195–202, 2020.

[60] P. J. Desai, Prof. A. K. Kadam, Prof. M. S. Bewoor, H. Mahmood, and A. A. Al-, “An Approach for Prediction of Obstructive Sleep Apnea,” vol. 8, no. 1, pp. 189–194, 2020.

[61] R. S. Suryawanshi, A. Kadam, and D. R. Anekar, “Software defect prediction: A survey with a machine learning approach,” Int. J. Adv. Sci. Technol., vol. 29, no. 5, pp. 330–335, 2020.

[62] A. Kurhade, J. Naveenkumar, and A. K. Kadam, “An experimental on top-k high utility itemset mining by efficient algorithm Tkowithtku,” Int. J. Innov. Technol. Explore. Eng., vol. 8, no. 8 Special Issue 3, pp. 519–522, 2019.

[63] A. A. Kore, D. M. Thakore, and A. K. Kadam, “Unsupervised extraction of common product attributes from E-commerce websites by considering client suggestion,” Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 11, pp. 1199–1203, 2019.

[64] Dr.S. D. Joshi, Dr. A. K. Kadam, Pritee Hulule, “A Survey Novel Approach for Efficient Selection of Test Case Prioritization Techniques,” vol. 3085, no. 12, pp. 999–1001, 2018.

[65] Dr. A. K. Kadam, Amruta Magdum, prof. Dr. S. D Joshi, “A Survey on Test Case Prioritization with Rate of Fault Detection,” Int. J. Res. Electron. Comput. Eng., vol. 6, no. 4, 2018.

[66] Dr. Vinod H Patil, Dr. Anurag Shrivastava, Devvret Verma, Dr. A L N Rao, Prateek Chaturvedi, Shaik Vaseem Akram, “Smart Agricultural System Based on Machine Learning and IoT Algorithm”, 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), 2022. DOI: DOI: 10.1109/ICTACS56270.2022.9988530

[67] Dr. Vinod H Patil, Dr. Pramod A Jadhav, Dr. C. Vinotha, Dr. Sushil Kumar Gupta, Bijesh Dhyani, Rohit Kumar,” Asset Class Market Investment Portfolio Analysis and Tracking”, 5th International Conference on Contemporary Computing and Informatics (IC3I), December 2022. DOI: 10.1109/IC3I56241.2022.10072525

[68] Dr. Vinod H Patil, Prasad Kadam, Sudhir Bussa, Dr. Narendra Singh Bohra, Dr. ALN Rao, Professor, Kamepalli Dharani,” Wireless Communication in Smart Grid using LoRa Technology”, 5th International Conference on Contemporary Computing and Informatics (IC3I), December 2022, DOI: 10.1109/IC3I56241.2022.10073338

[69] Vinod H. Patil, Dr Shruti Oza, Vishal Sharma, Asritha Siripurapu, Tejaswini Patil, “A Testbed Design of Spectrum Management in Cognitive Radio Network using NI USRP and LabVIEW”, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2S8, August 2019.

[70] Vinod H. Patil, Shruti Oza, “Green Communication for Power Distribution Smart Grid”, International Journal of Recent Technology and Engineering™ (IJRTE), ISSN:2277-3878(Online), Reg. No.: C/819981, Volume-8, Issue-1, Page No. 1035-1039, May-19.

[71] Patil, V.H., Oza, S., Sharma, V., Siripurapu, A., Patil, T.,” A testbed design of spectrum management in cognitive radio network using NI USRP and LabVIEW”, International Journal of Innovative Technology and Exploring Engineering, 2019, 8(9 Special Issue 2), pp. 257–262

[72] Vinod Patil et al, “A Model Design of Green Communication for Smart Grid Systems” SSRG International Journal of Electrical and Electronics Engineering, ISSN: 2348-8379, Volume 10 Issue 5, 227-239, May 2023. https://doi.org/10.14445/23488379/IJEEE-V10I5P121

[73] George, N. ., & B. K., A. . (2023). Hypervolume Sen Task Scheduilng and Multi Objective Deep Auto Encoder based Resource Allocation in Cloud.

International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 16–27. https://doi.org/10.17762/ijritcc.v11i4s.6303

[74] Renato Costa, Deep Reinforcement Learning for Autonomous Robotics , Machine Learning Applications Conference Proceedings, Vol 2 2022.

[75] Dhabliya, D. (2019). Security analysis of password schemes using virtual environment. International Journal of Advanced Science and Technology, 28(20), 1334-1339. Retrieved from www.scopus.com