International Journal of Communication Networks and Information Security
2024, 16(4)
ISSN: 2073-607X,2076-0930
https://ijcnis.org/

Research Article

Analyzing the Behavior of Software Reliability Execution Time Models for Different Agile–Scrum Based Projects

1stDr. Nikhil Govil [1*], 2ndDr. Rinku Sharma Dixit [2], 3rdDr. Shailee Lohmor Choudhary [3]

[1]Associate Professor,Department of CEA, GLA University, Mathura, U.P., India
[2]Professor, Department of AI/ML – Data Sciences, New Delhi Institute of Management, New Delhi, India
[3] Associate Professor, Department of AI/ML – Data Sciences, New Delhi Institute of Management, New Delhi, India

***Corresponding Author:**rinku.dixit@ndimdelhi.org

Citation: Dr. Nikhil Govil, "Analyzing the Behavior of Software Reliability Execution Time Models for Different Agile–Scrum Based Projects," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 16, no 4, 2024. pp.587-601.

ARTICLE INFO

Received:
Accepted:

ABSTRACT

Software reliability is an essential component of software development stages. The reliability of a software system plays a vital role in the overall development and success of that software. Reliability is broad and linked to other areas of technologies and approaches. This requires a proactive mechanism, which includes not only technical aspects but also legal and ethical considerations. Maintaining reliability in Agile-based software is an arduous exercise. This happens because in an Agile-based project, frequent changes in requirements are expected during the development cycle. To develop high levels of reliability in an Agile environment, the Quality Assurance (QA) engineer needs to carefully select the appropriate reliability model. In this research paper, we studied the performance of two most popular reliability models namely Basic Execution Time Model and Logarithmic Poisson Execution Model. Since the basic idea behind developing software is quite different in an Agile environment compared to traditional software development processes; a dataset of 30 Agile-based projects has been prepared for the purposes of calculation. These 30 projects in this dataset are divided into three groups as low, medium and high-level projects based on the number of sprints required to complete the work. This paper presents a comparative analysis of these two reliability models on various parameters. As a result, we found that the Logarithmic Poisson Execution Model produces optimal results for most Agile-based projects in all 3 project categories.

Keywords: Agile methodologies; Scrum; Software reliability; Quality assurance; Software failure intensity

INTRODUCTION

In today's era, many software are being developed with the help of Agile Methodology in software industries. There are many frameworks under Agile Methodologies, almost all of which give great importance to the suggestions and requirements of their customers. Developing projects as per customer requirements is much easier in Agile methodologies [1]. This is possible because there are a total of twelve principles and four values mentioned in the Agile Manifesto of Agile Methodologies on which professionals have been continuously working. It is very beneficial [2] to follow the Agile Manifesto to develop software in an efficient manner.

Developing software with comparatively more efficiency and with minimum effort is a big challenging task for professionals. Just by developing a software, the work of the software development team does not end. It is the responsibility of software developers to ensure efficiency as well as security, quality, maintainability, adaptability,

robustness etc. in any software. Along with this, there is another very essential element, which is reliability. Quality and reliability are often considered the same [3]. But in reality, these two are different components which have their own importance in software.

While on one hand quality means 'fitness for use', on the other hand reliability means 'ability to perform for a specified period of time without failure'. There are many models available to measure and ensure reliability in software, using which the software system can be made more reliable. Maintaining reliability is very challenging in an Agile environment [4], where rapid changes occur frequently. Developing reliable software is one of the most uphill tasks in Agile software development [5], reliability is measured by customer feedback in the form of defects.

In this research paper we have discussed the requirements and limitations of software reliability models in the context of Agile software. A software reliability model indicates the form of a stochastic process that defines the behavior of software failures over time. There are many models in practice to measure software reliability, but there is no reliability model dedicated to Agile methodologies. In order to explore the possibilities of software reliability models in Agile methodologies, in our research paper we have mainly tested two reliability models on the data of different projects. Based on the calculations of both these models, a comparative study of these models has been presented

LITERATUREREVIEW

In this section, we provide a literature survey for some published research works in the domain of software reliability related to Agile methodologies.

S. Dave et al. [6] discussed the widespread use of machine learning in their research paper. But despite its widespread use, security and hardware failures still remain a challenge in Agile systems. In their article, the authors have presented the concept of a reliable and relatively more secure machine learning system as a solution to these problems. In [7], the authors highlighted the challenges associated with the sprint planning process. These challenges can significantly impact the reliability of Sprint delivery in the future. In their research paper, authors have shown three major challenges and suggested guidelines to reduce their impact.

In the research paper [8], the authors have considered software reliability as a serious issue during Agile testing. For their research work, the authors conducted their tests on Jelinski–Moranda model and Goel–Okumoto model. As a result of the research, the authors have shown accurate prediction of software reliability for Agile testing environments. In the research paper [9], the authors have worked towards increasing customer satisfaction and maximizing feedback. Along with this, the help of adaptive neuro-fuzzy estimation system has been taken to implement the uncertain demands of the customers without any disruption. A new multi-objective model focusing on the lot-sizing and scheduling problem has been proposed to improve reliability in Agile software.

In [10], the authors have mentioned that Agile methodologies are being adopted very rapidly in the IT industries in the last few years. In such a situation, it is important to focus more on reliability and security in Agile based software. But sometimes some obstacles have to be faced while adopting security systems. In their research paper, to eliminate these obstacles, the authors have proposed a new Agile method, which they have proposed on the basis of railway domain. The authors [11] show that reliability has a huge impact in software systems. Software reliability is a very important factor in determining the quality of software. In their research paper, they have presented estimates of reliability and an assessment of their methods. The authors provide a critical analysis of effective testing in dependency assessment.

Expressing concern over potential reliability issues in the medical world, the authors [12] have proposed a new software reliability development model. When this proposed model was compared with the NHPP SRMEs model, it was found that the newly proposed model provides more accurate analysis. In their research work, the authors [13] have studied about 200 other research papers. In their survey, the authors present a classification of various methods proposed in the literature to predict software reliability. With the application of which software can be developed more efficiently and qualitatively.

The authors [14] noted that software reliability is an essential component of ensuring the quality and dependability of a software system. Accurate fault resolution and independent fault resolution are two factors on which there is still great potential in research. In their research paper, the authors have given an introduction related to fault-dependent detection, incomplete fault resolution and faults present in the system. In this paper [15], the authors point out that traditional software reliability enhancement models often only consider fault

detection data. Whereas quality, safety etc. are other elements on which research is equally important. The authors considered data from a NASA project to develop differential equation-based models for fault detection and resolution

METHODOLOGY

Software reliability refers to the ability of a software system to continuously perform its intended functions without failure under specified conditions for a specified period of time. This is an important quality characteristic of software, especially in applications where failure can have serious consequences, such as in aerospace, healthcare, automotive, and financial systems. The software reliability model [16] specifies the general form of the dependence of the failure process on the factors such as, fault introduction, fault removal, and the environment.

In this research paper, we have presented a comparative study of two major software reliability models [17], Basic Execution Time Model and Logarithmic Poisson Execution Time Model. This study has been done on a data set of 30 Agile-based projects. These data sets have been divided into three parts. The first part consists of low-level projects in which the number of required sprints is maximum 10. The second part consists of medium-level projects in which the number of required sprints ranges from 11 to 19. The third and final part consists of high-level projects in which the number of required sprints ranges from 20 to 29.

Basic Execution Time Model

J.D. Musa created the fundamental execution time model in 1979. Failures could happen in accordance with a Non-Homogeneous Poisson Process (NHPP), according to this concept [18]. Nonetheless, the decline in failure intensity is consistent when considering the quantity of failures detected. This can be represented by the following formula:

$$\square(\mu) = \square_0[1 - (\mu/V_0)] \qquad (1)$$

Where,

$\square$: # of failures intensity at the start of the execution

$\square$: average or expected # of failures experienced at a given point in time. V_0: # of failures experienced (when program is executed for $\square$ time period).

The relationship between failure intensity ($\square$) and mean failures experienced ($\square$) is represented in following figure:

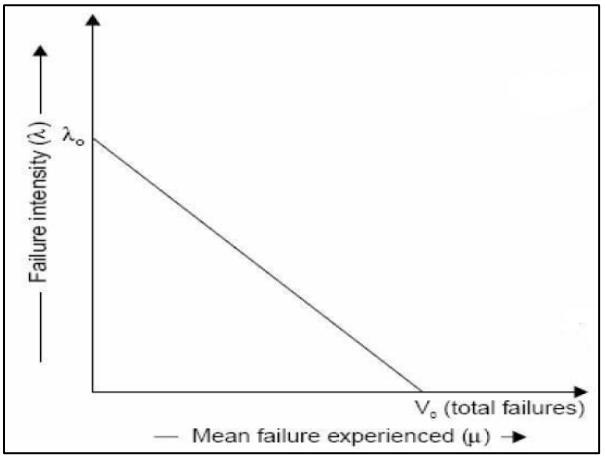

Figure 1: Failure intensity ($\square$) as a function of $\square$ for basic model

The basic execution time model in software reliability focuses on estimating the time it takes for a software program to execute successfully without failing or encountering errors. Unlike the previous basic execution time model, which focuses on estimating the actual execution time of a program, this model is centered around predicting how long a software system can run reliably without encountering failures or errors. The basic execution time model in software reliability focuses on estimating the reliability and availability of a software system over time. It considers both the ability to operate without errors (MTBF) and the ability to recover quickly from errors (MTTR), along with factors like redundancy, fault tolerance, and error recovery mechanisms. Accurate estimation and measurement of these parameters are crucial for ensuring that software systems meet

reliability requirements.

Logarithmic Poisson Execution Time Model

J.D. Musa et al. also developed LPETM (The Logarithmic Poisson Execution Time Model). The function for failure intensity is different in this model [18] as compared to the Basic Model.

Here, failure intensity function (decrement per failure) decreases exponentially whereas it is constant for basic model. The failure intensity function can be represented as:

$$\square(\mu) = \square_0 \exp(-\square\mu) \quad (2)$$

Where,

$\square$: failure intensity decay parameter

The relationship between failure intensity ($\square$) and mean failures experienced ($\square$) can be demonstrated as following figure:

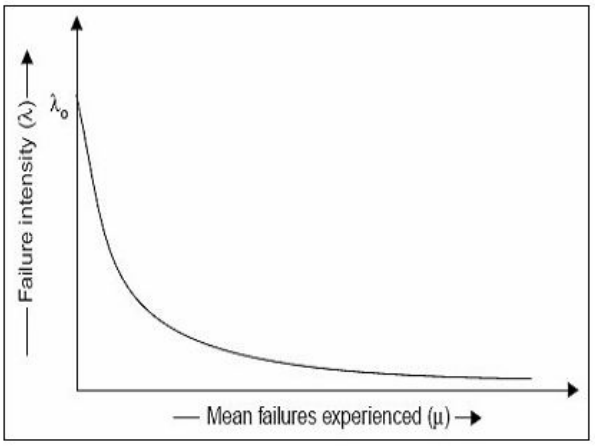

Figure 2: Relationship between □ and □

LPETM is a mathematical model used in software reliability engineering to estimate software reliability by analyzing fault detection and removal processes over time. This model takes into account both the fault removal process (detection and correction of defects) and the software reliability growth process. The LPETM is an improvement over earlier models like the NHPP (Non-Homogeneous Poisson Process) model. Using the LPETM, software reliability engineers can estimate the software's reliability over time, calculate metrics like the Mean Time Between Failures (MTBF), and make predictions about future reliability based on the fault arrival rate, fault detection rate, and defect removal efficiency. The goal is to improve software reliability by monitoring and optimizing the fault removal process.

Data Sets

In this section, two different datasets are taken separately for the Basic Execution Time Model and Logarithmic Poisson Execution Time model. These datasets consist of 30 individual Agile-based projects ranging from PR1 to PR30 as low, medium, and high-level project factors, respectively. Due to the unavailability of actual and real-world project's data over the web; we have prepared these datasets for both the models. However, these datasets are further validated from industry experts who have rich experience of developing Agile-based projects.

Dataset for Basic Execution Time Model

In this dataset, we have taken inputs as total number of sprints required, initial failure intensity ($\square_0$), total number of failures experienced (V_0), average or expected number of failures expected (μ), execution time ($\square$), and failure intensity objective ($\square_F$). These inputs are individually taken for low, medium and high-level project groups.

Table 1: Dataset inputs for low, medium and high-level projects as per Basic Execution Time Model.

Low										
Inputs	**PR1**	**PR2**	**PR3**	**PR4**	**PR5**	**PR6**	**PR7**	**PR8**	**PR9**	**PR10**
No. of Sprints required	6	9	7	9	8	6	7	10	6	7
Initial failure intensity ($\square_0$)	15	16	14	16	15	14	15	16	14	14
No. of failures experienced (V_0)	116	125	117	126	123	114	118	127	108	112
Avg. or expected no. of failures expected ($\square$)	56	58	48	57	56	51	52	55	51	50
Execution time ($\square$)	15	15	15	15	15	15	15	15	15	15
Failure intensity objective ($\square_F$)	3	3	3	3	3	3	3	3	3	3
Medium										
Inputs	**PR11**	**PR12**	**PR13**	**PR14**	**PR15**	**PR16**	**PR17**	**PR18**	**PR19**	**PR20**
No. of Sprints required	12	17	13	18	15	11	12	19	12	13
Initial failure intensity ($\square_0$)	20	23	19	22	21	20	19	22	20	21
No. of failures experienced (V_0)	200	218	202	217	214	198	204	217	193	196
Avg. or expected no. of failures expected ($\square$)	100	105	98	104	105	99	101	103	98	98
Execution time ($\square$)	20	20	20	20	20	20	20	20	20	20
Failure intensity objective ($\square_F$)	5	5	5	5	5	5	5	5	5	5
High										
Inputs	**PR21**	**PR22**	**PR23**	**PR24**	**PR25**	**PR26**	**PR27**	**PR28**	**PR29**	**PR30**
No. of Sprints required	23	28	24	29	26	21	22	20	23	25
Initial failure intensity ($\square_0$)	35	39	33	38	36	34	34	38	35	37
No. of failures experienced (V_0)	308	335	339	361	342	317	319	320	334	341
Avg. or expected no. of failures expected ($\square$)	154	160	153	159	257	153	154	151	156	155
Execution time ($\square$)	35	35	35	35	35	35	35	35	35	35
Failure intensity objective ($\square_F$)	8	8	8	8	8	8	8	8	8	8

Logarithmic Poisson Execution Time Model

In this dataset, we have taken inputs as total number of sprints required, initial failure intensity ($\square_0$), failure intensity decay parameter ($\square$), experienced failures (μ), failure experienced and failure intensity after CPU hour ($\square$), and failure intensity objective ($\square_F$). These inputs are individually taken for low, medium and high-level project groups.

Table 2: Dataset inputs for low, medium and high-level projects as per Logarithmic Poisson Execution Time Model.

Low										
Inputs	**PR1**	**PR2**	**PR3**	**PR4**	**PR5**	**PR6**	**PR7**	**PR8**	**PR9**	**PR10**
No. of Sprints required	6	9	7	9	8	6	7	10	6	7
Initial failure intensity (λ_0)	4	6	4	6	5	4	5	6	5	4
Failure intensity decay parameter (θ)	0.01	0.02	0.01	0.02	0.02	0.01	0.01	0.02	0.01	0.01
Experienced failures (μ)	58	43	62	42	39	57	78	46	77	59
Failure experienced & failure intensity after CPU hour (τ)	6	6	6	6	6	6	6	6	6	6
Failure intensity objective (λ_F)	2	2	2	2	2	2	2	2	2	2
Medium										
Inputs	**PR11**	**PR12**	**PR13**	**PR14**	**PR15**	**PR16**	**PR17**	**PR18**	**PR19**	**PR20**
No. of Sprints required	12	17	13	18	15	11	12	19	12	13
Initial failure intensity (λ_0)	20	23	19	22	21	20	19	22	20	21
Failure intensity decay parameter (θ)	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Experienced failures (μ)	100	105	98	104	105	99	101	103	98	98
Failure experienced & failure intensity after CPU hour (τ)	20	20	20	20	20	20	20	20	20	20
Failure intensity objective (λ_F)	2	2	2	2	2	2	2	2	2	2
High										
Inputs	**PR21**	**PR22**	**PR23**	**PR24**	**PR25**	**PR26**	**PR27**	**PR28**	**PR29**	**PR30**
No. of Sprints required	23	28	24	29	26	21	22	20	23	25
Initial failure intensity (λ_0)	30	33	33	38	36	34	34	38	35	37
Failure intensity decay parameter (θ)	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Experienced failures (μ)	115	114	113	119	107	113	116	112	108	116
Failure experienced & failure intensity after CPU hour (τ)	35	35	35	35	35	35	35	35	35	35
Failure intensity objective (λ_F)	2	2	2	2	2	2	2	2	2	2

Result Analysis and Discussion

In this section, we provide result performed on the datasets for both the models. A detailed discussion is also provided for better analysis and understanding. Table 3 depicts the results of Basic Execution Time model based on Table 2. Here as an output, current failure intensity [□(□)], additional failure required to be experienced to reach failure intensity objective (□μ), and additional time required to reach the objective (□□) for 3 failures/CPU hour for low-level projects, 5 failures/CPU for medium-level projects, and 8 failures/CPU for high-level projects respectively.

Table 3: Computation of results of Basic Execution Time Model

Low										
Outputs	**PR1**	**PR2**	**PR3**	**PR4**	**PR5**	**PR6**	**PR7**	**PR8**	**PR9**	**PR10**
□(□)	7.76	8.58	8.26	8.76	8.17	7.74	8.39	9.07	7.39	7.75
d□/d□	-0.13	-0.13	-0.12	-0.13	-0.12	-0.12	-0.13	-0.13	-0.13	-0.13
□(□) [after 15 CPU hour]	107.26	115.34	106.31	116.06	112.27	104.22	108.72	116.78	99.92	102.81
□(□) [after 15 CPU hour]	1.13	1.24	1.28	1.26	1.31	1.20	1.18	1.29	1.05	1.15
□□ [3 failures /CPU hour]	21.33	27.94	27.21	29.63	26.00	22.29	26.67	32.31	18.43	22.00
□□ [3 failures /CPU hour]	3.40	4.22	4.19	4.42	4.03	3.56	4.07	4.73	3.01	3.51
Medium										
Outputs	**PR11**	**PR12**	**PR13**	**PR14**	**PR15**	**PR16**	**PR17**	**PR18**	**PR19**	**PR20**
□(□)	10.00	11.92	9.78	11.46	10.70	10.00	9.59	11.56	9.84	10.50
d□/d□	-0.10	-0.11	-0.09	-0.10	-0.10	-0.10	-0.09	-0.10	-0.10	-0.11
□(□) [after 20 CPU hour]	172.93	191.57	171.21	188.43	183.94	171.74	172.33	188.43	168.71	173.01
□(□) [after 20 CPU hour]	2.71	2.79	2.90	2.90	2.95	2.65	2.95	2.90	2.52	2.46
□□ [5 failures /CPU hour]	50.00	65.61	50.84	63.68	58.05	49.50	49.32	64.68	46.75	51.33
□□ [5 failures /CPU hour]	6.93	8.24	7.14	8.18	7.75	6.86	7.00	8.27	6.54	6.93
High										
Outputs	**PR21**	**PR22**	**PR23**	**PR24**	**PR25**	**PR26**	**PR27**	**PR28**	**PR29**	**PR30**
□(□)	17.50	20.37	18.11	21.26	8.95	17.59	17.59	20.07	18.65	20.18
d□/d□	-0.11	-0.12	-0.10	-0.11	-0.11	-0.11	-0.11	-0.12	-0.10	-0.11
□(□) [after 35 CPU hour]	302.23	329.31	327.77	351.93	333.41	309.57	311.35	314.99	325.47	333.35
□(□) [after 35 CPU hour]	0.66	0.66	1.09	0.95	0.90	0.80	0.82	0.60	0.89	0.83
□□ [8 failures /CPU hour]	83.60	106.28	103.82	126.00	9.00	89.41	89.94	101.63	101.66	112.27

□□ [8 failures /CPU hour]	6.89	8.03	8.39	9.29	1.06	7.35	7.39	7.75	8.08	8.53

Table 4 depicts the results of Logarithmic Poisson Execution Time model based on Table 2. Here as an output, current failure intensity [□(□)], additional failure required to be experienced to reach failure intensity objective (□μ), and additional time required to reach the objective (□□) for 2 failures/CPU hour for low-level, medium-level& high-level projects.

Table 4: Computation of results of Logarithmic Poisson Execution Time Model

Low										
Outputs	**PR1**	**PR2**	**PR3**	**PR4**	**PR5**	**PR6**	**PR7**	**PR8**	**PR9**	**PR10**
□(□)	2.24	2.54	2.15	2.59	2.29	2.26	2.29	2.39	2.32	2.22
d□/d□	-0.02	-0.05	-0.02	-0.05	-0.05	-0.02	-0.02	-0.05	-0.02	-0.02
□(□) [after 6 CPU hour]	21.52	27.12	21.52	27.12	23.50	21.52	26.24	27.12	26.24	21.52
□(□) [after 6 CPU hour]	3.23	3.49	3.23	3.49	3.13	3.23	3.85	3.49	3.85	3.23
□□ [2 failures /CPU hour]	11.32	11.93	7.32	12.93	6.82	12.32	13.63	8.93	14.63	10.32
□□ [2 failures /CPU hour]	5.35	5.31	3.53	5.70	3.19	5.79	6.37	4.09	6.80	4.90
Medium										
Outputs	**PR11**	**PR12**	**PR13**	**PR14**	**PR15**	**PR16**	**PR17**	**PR18**	**PR19**	**PR20**
□(□)	2.71	2.82	2.68	2.75	2.57	2.76	2.52	2.80	2.82	2.96
d□/d□	-0.05	-0.06	-0.05	-0.05	-0.05	-0.06	-0.05	-0.06	-0.06	-0.06
□(□) [after 20 CPU hour]	109.88	116.14	107.61	114.14	112.06	109.88	107.61	114.14	109.88	112.06
□(□) [after 20 CPU hour]	2.22	2.25	2.21	2.24	2.23	2.22	2.21	2.24	2.22	2.23
□□ [2 failures /CPU hour]	15.13	17.12	14.57	15.90	12.57	16.13	11.57	16.90	17.13	19.57
□□ [2 failures /CPU hour]	6.53	7.25	6.32	6.81	5.56	6.89	5.16	7.17	7.25	8.10
High										
Outputs	**PR21**	**PR22**	**PR23**	**PR24**	**PR25**	**PR26**	**PR27**	**PR28**	**PR29**	**PR30**
□(□)	3.01	3.38	3.44	3.52	4.24	3.55	3.34	4.05	4.04	3.64
d□/d□	-0.06	-0.07	-0.07	-0.07	-0.08	-0.07	-0.07	-0.08	-0.08	-0.07
□(□) [after 35 CPU hour]	154.58	159.14	159.14	165.92	163.32	160.57	160.57	165.92	161.96	164.64
□(□) [after 35 CPU hour]	1.36	1.37	1.37	1.38	1.37	1.37	1.37	1.38	1.37	1.38
□□ [2 failures /CPU	20.41	26.17	27.17	28.23	37.53	28.67	25.67	35.23	35.12	29.89

hour]										
□□ [2 failures /CPU hour]	6.53	7.25	6.32	6.81	5.56	6.89	5.16	7.17	7.25	8.10

Comparison for Low Level Projects

Figure 3 depicts the Comparison of Current Failure Intensity for low-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low Current Failure Intensity as compared to the Basic Execution Time Model for all 10 Agile projects. Here, the lowest and highest values for current failure intensity in Basic model are 7.39 and 9.07 respectively. While the lowest and highest values for current failure intensity in Logarithmic model are 2.15 and 2.59 respectively.

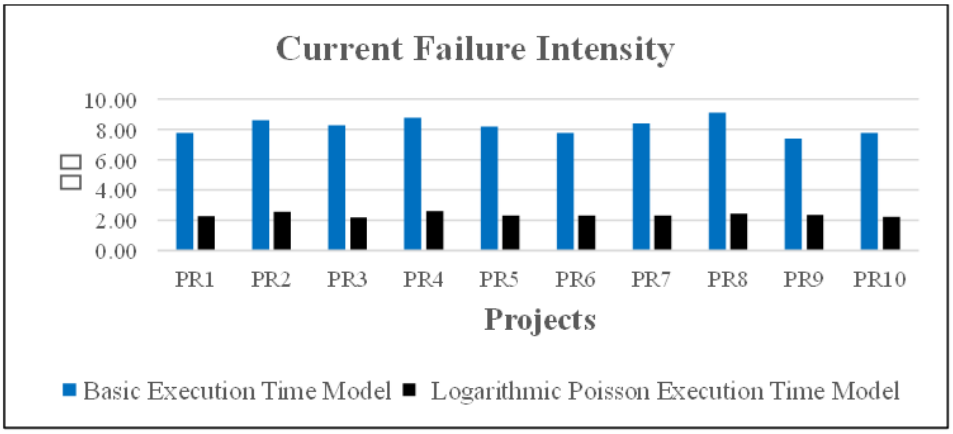

Figure 3: Comparison of Current Failure Intensity for low level projects

Figure 4 depicts the Comparison of Additional failure required to be experienced to reach failure intensity objective for low-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low additional failure required to be experienced to reach failure intensity objective as compared to the Basic Execution Time Model for all 10 Agile projects. Here, the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Basic model are 18.43 and 32.31 respectively. While the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 6.82 and 14.63 respectively.

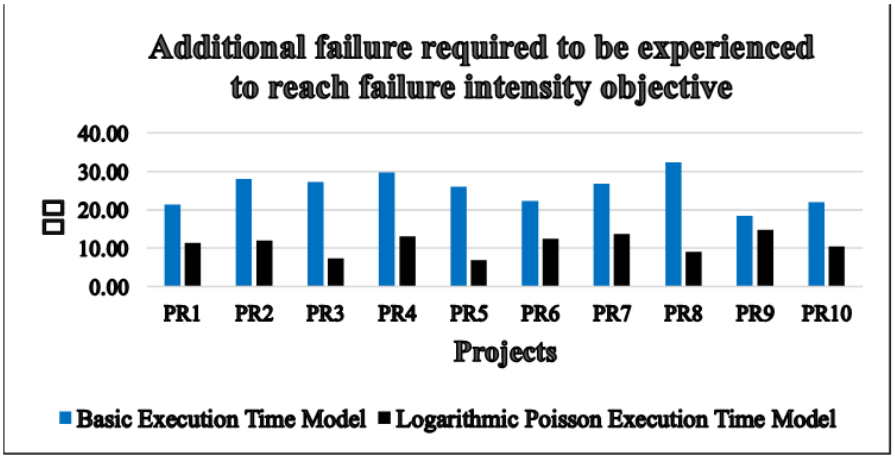

Figure 4: Comparison of Additional failure required to be experienced to reach failure intensity objective for low level projects

Figure 5 depicts the Comparison of Additional time required to reach the objective for low-level projects. As per the graph, Logarithmic Poisson Execution Time Model have high additional time required to reach the

objective as compared to the Basic Execution Time Model. Here, the lowest and highest values for additional time required to reach the objective in Basic model are 3.01 and 4.73 respectively. While the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 3.19 and 6.80 respectively.

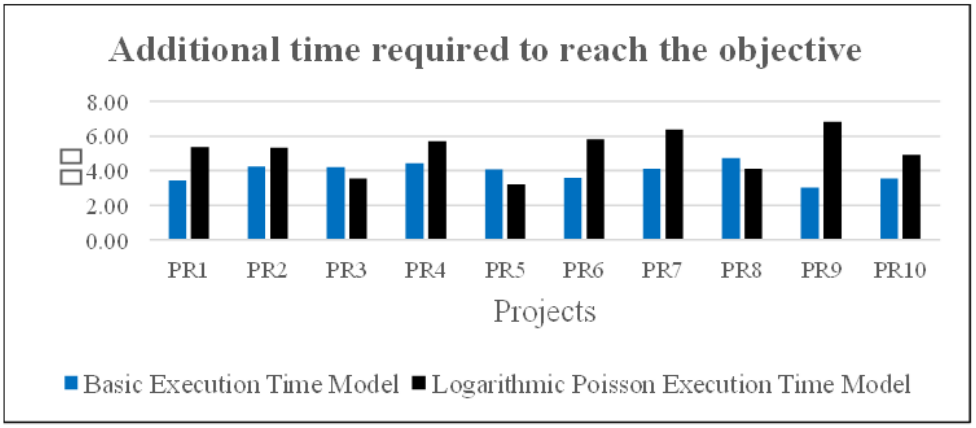

Figure 5: Comparison of Additional time required to reach the objective for low level projects

Comparison for Medium Level Projects

Figure 6 depicts the Comparison of Current Failure Intensity for medium-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low Current Failure Intensity as compared to the Basic Execution Time Model for all 10 Agile projects. Here, the lowest and highest values for current failure intensity in Basic model are 9.59 and 11.92 respectively. While the lowest and highest values for current failure intensity in Logarithmic model are 2.52 and 2.96 respectively.

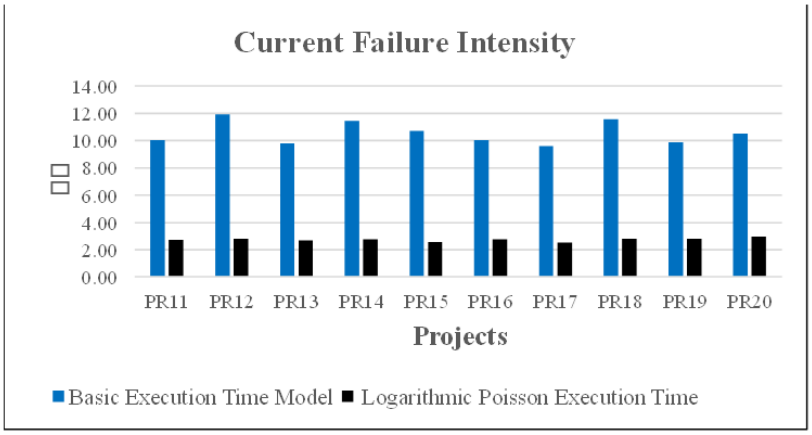

Figure 6: Comparison of Current Failure Intensity for medium level projects

Figure 7 depicts the Comparison of Additional failure required to be experienced to reach failure intensity objective for medium-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low additional failure required to be experienced to reach failure intensity objective as compared to the Basic Execution Time Model for all 10 Agile projects. Here, the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Basic model are 46.75 and 65.61 respectively. While the

lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 11.57 and 19.57 respectively.

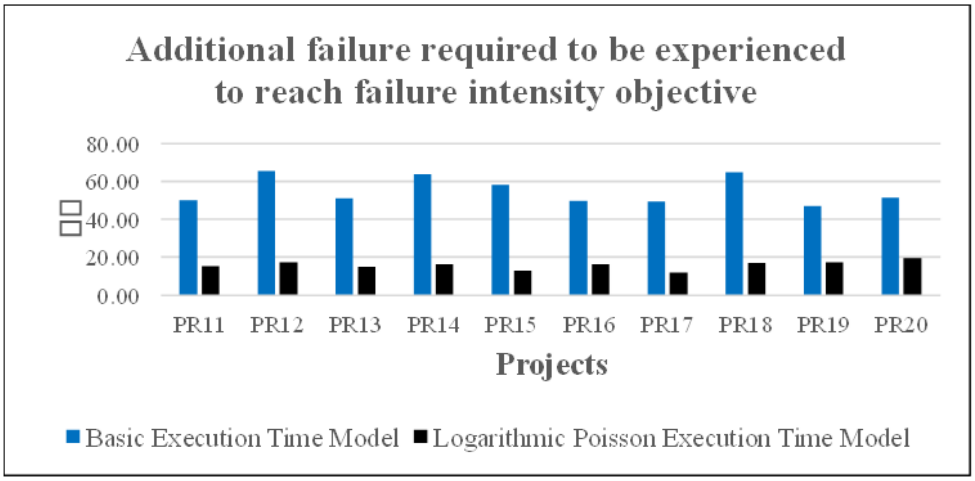

Figure 7: Comparison of Additional failure required to be experienced to reach failure intensity objective for medium level projects

Figure 8 depicts the Comparison of Additional time required to reach the objective for mediumlevel projects. As per the graph, Logarithmic Poisson Execution Time Model has low additional time required to reach the objective as compared to the Basic Execution Time Model for most of the Agile projects. Here, the lowest and highest values for additional time required to reach the objective in Basic model are 6.54 and 8.27 respectively. While the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 5.16 and 8.10 respectively.

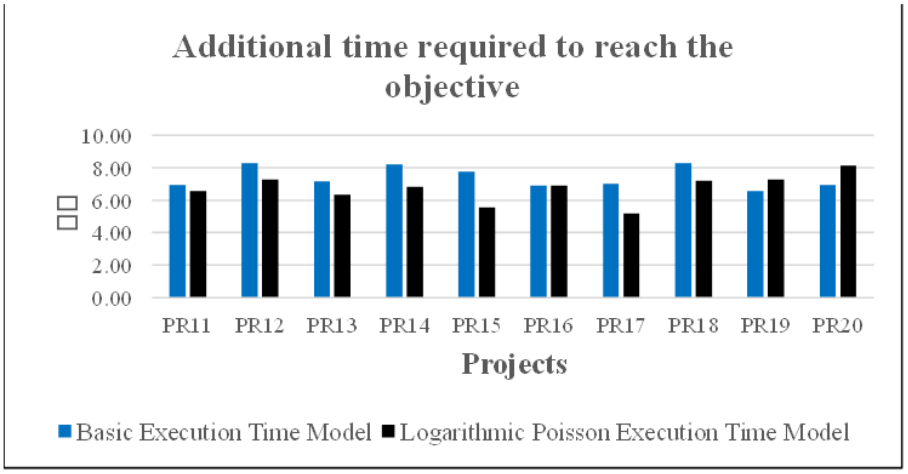

Figure 8: Comparison of Additional time required to reach the objective for medium level projects

Comparison for High Level Projects

Figure 9 depicts the Comparison of Current Failure Intensity for high-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low Current Failure Intensity as compared to the Basic Execution Time Model for all 10 Agile projects. Here, the lowest and highest values for current failure intensity in Basic model are 8.95 and 21.26 respectively. While the lowest and highest values for current failure intensity in Logarithmic model are 3.01 and 4.24 respectively.

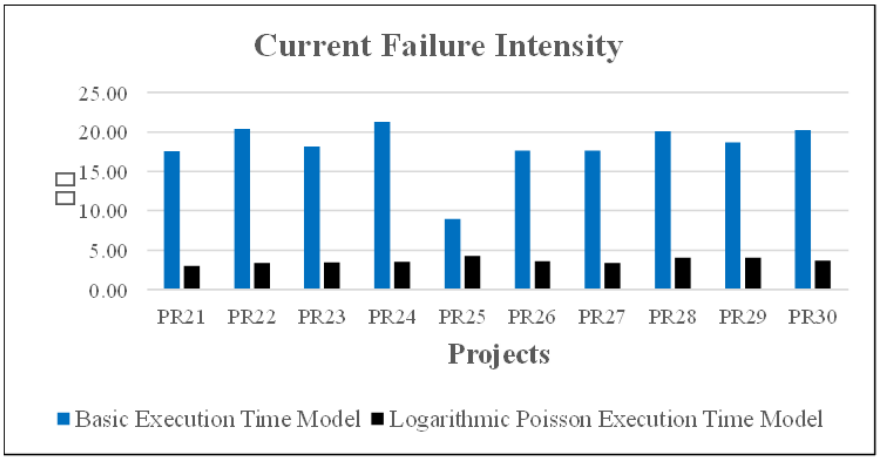

Figure 9: Comparison of Current Failure Intensity for high level projects

Figure 10 depicts the Comparison of Additional failure required to be experienced to reach failure intensity objective for high-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low additional failure required to be experienced to reach failure intensity objective as compared to the Basic Execution Time Model for most of the Agile projects. Here, the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Basic model are 9.00 and 112.27 respectively. While the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 20.41 and 37.53 respectively.

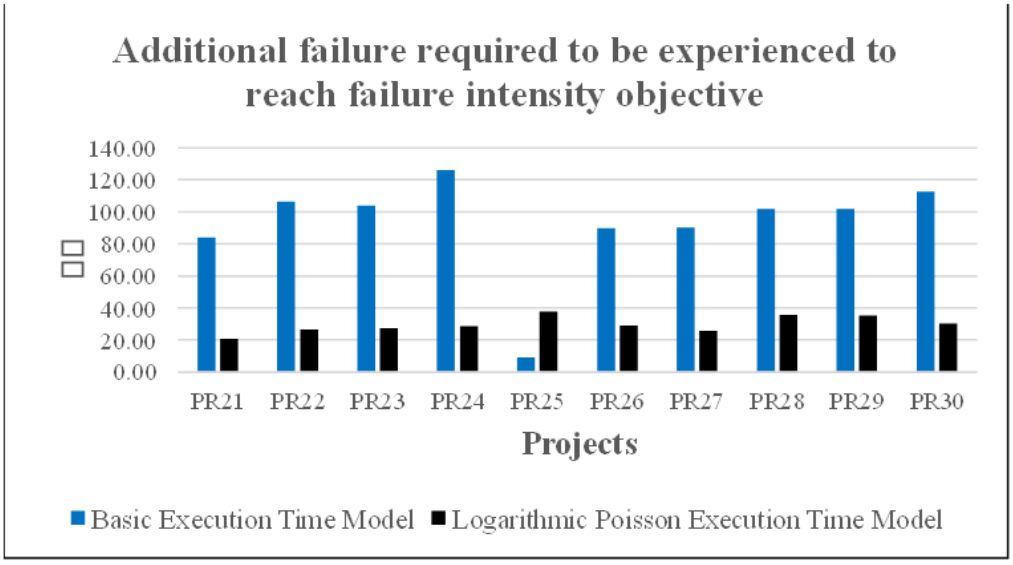

Figure 10: Comparison of Additional failure required to be experienced to reach failure intensity objective for high level projects

Figure 11 depicts the Comparison of Additional time required to reach the objective for high-level projects. As per the graph, Logarithmic Poisson Execution Time Model has low additional time required to reach the objective as compared to the Basic Execution Time Model for most of the Agile projects. Here, the lowest and highest values for additional time required to reach the objective in Basic model are 1.06 and 9.29 respectively. While the lowest and highest values for additional failure required to be experienced to reach failure intensity objective in Logarithmic model are 5.16 and 8.10 respectively.

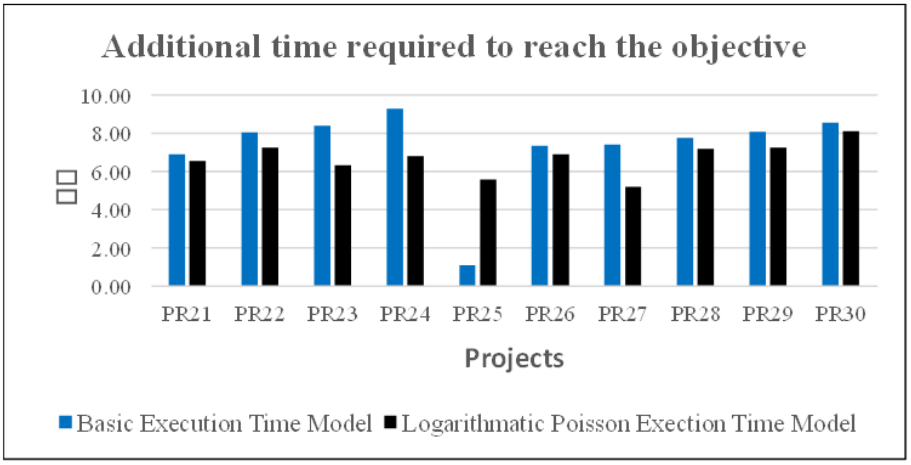

Figure 11: Comparison of Additional time required to reach the objective for high level projects

Difference between both Execution Time Models

In this subsection, we provided the project wise difference among both the models for low-level, medium-level and high-level projects in Table 5.

Table 5: Project wise difference among both the models for low-level, medium-level and high-level projects

Low Level Projects										
Projects→	**PR1**	**PR2**	**PR3**	**PR4**	**PR5**	**PR6**	**PR7**	**PR8**	**PR9**	**PR10**
□(□)	5.52	6.04	6.10	6.17	5.88	5.47	6.10	**6.68**	**5.07**	5.53
□□	10.02	16.00	19.90	16.69	19.18	9.97	13.04	**23.38**	**3.80**	11.68
□□	-1.95	-1.09	0.67	-1.28	**0.84**	-2.24	-2.30	0.64	**-3.79**	-1.39
Medium Level Projects										
Projects→	**PR11**	**PR12**	**PR13**	**PR14**	**PR15**	**PR16**	**PR17**	**PR18**	**PR19**	**PR20**
□(□)	7.29	**9.11**	7.11	8.71	8.12	7.24	7.07	8.75	**7.03**	7.54
□□	34.87	**48.49**	36.27	47.78	45.48	33.37	37.75	47.78	**29.62**	31.76
□□	0.41	0.99	0.82	1.37	**2.19**	-0.03	1.84	1.10	-0.71	**-1.17**
High Level Projects										
Projects→	**PR21**	**PR22**	**PR23**	**PR24**	**PR25**	**PR26**	**PR27**	**PR28**	**PR29**	**PR30**
□(□)	14.49	17.00	14.66	**17.75**	**4.71**	14.04	14.24	16.02	14.62	16.55
□□	63.19	80.11	76.65	**97.77**	**-28.53**	60.75	64.28	66.40	66.54	82.38
□□	0.36	0.78	2.07	**2.48**	**-4.49**	0.45	2.23	0.58	0.83	0.43

For Current Failure Intensity [□(□)], the minimum difference is 5.07 and maximum difference is 6.68 for low-level projects. The minimum difference is 7.03 and maximum difference is 9.11 for medium-level projects. The minimum difference is 4.71 and maximum difference is 17.75 for high-level projects.

For Additional failure required to be experienced to reach failure intensity objective (□□), the minimum difference is 3.80 and maximum difference is 23.38 for low-level projects. The minimum difference is 29.62 and maximum difference is 48.49 for medium-level projects. The minimum difference is (– 28.53) and maximum difference is 97.77 for high-level projects.

For Additional time required to reach the objective (□□), the minimum difference is (– 3.79) and maximum difference is 0.84 for low-level projects. The minimum difference is (– 1.17) and maximum difference is 2.19 for

medium-level projects. The minimum difference is (– 4.49) and maximum difference is 2.48 for high-level projects.

CONCLUSION AND FUTURE SCOPE

The criteria and their impact on the selection of relevant software reliability models is a very important step. However, there is not enough information in the literature about how to select an appropriate reliability model for Agile-based software systems. Therefore, there is a need to apply some mathematical analysis to finalize the optimal software reliability model that works efficiently on Agile-based software systems.

In this article, we selected two such software reliability models from traditional software development which are mainly used during software development in IT industries and with the help of which the chances of the software being reliable and quality-oriented increases to a great extent. These models are Basic Execution Time Model and Logarithmic Poisson Execution Model. Now the challenge before us was to demonstrate the usefulness of both these reliability models in an Agile environment. For this, we prepared a dataset of 30 Agile projects and presented a comparative study of both the models on the basis of sprints. During result analysis, we found that the Logarithmic Poisson Execution Model is providing optimum value in most Agile projects. Which meant that this model can be adopted by software industries for Agile projects. As a result, Agile based projects can now be developed more reliably than before.

However, through this research paper we have provided detailed computations and analysis of a dataset of 30 Agile projects on two different software reliability models. But we still feel there is room for future implementation and analysis. The Jelinski-Moranda Model, the Calendar Time Component and the Bug Seeding Model are other software reliability models that can be used with this dataset of Agile projects to analyze the behavior of Agile Projects. Also, a comparative study of all these models can be presented so that professionals working on Agile methodologies can select the appropriate model for software reliability on the basis of sprints and implement it in their project.

REFERENCES

[1] Ciancarini P, Ergasheva S, Farina M, Mubarakshin D and Succi G (2023) Agile methodologies between software development and music production: an empirical study. Front. Comput. Sci. 5:1181041. doi: 10.3389/fcomp.2023.1181041

[2] Mishra, A., Alzoubi, Y.I. Structured software development versus agile software development: a comparative analysis. Int J Syst Assur Eng Manag 14, 1504–1522 (2023). https://doi.org/10.1007/s13198-023-01958-5

[3] Ahmed M, Khan SUR, Alam KA (2023) An NLP-based quality attributes extraction and prioritization framework in agile-driven software development. Autom Softw Eng 30(1):7

[4] Dursun M, Goker N (2022) Evaluation of project management methodologies success factors using fuzzy cognitive map method: waterfall, agile, and lean six sigma cases. Int J Intell Syst Appl Eng 10(1):35–43

[5] Lee, W.-T.; Chen, C.-H. Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering. Electronics 2023, 12, 3291. https://doi.org/10.3390/electronics12153291

[6] S. Dave et al., “Special Session: Towards an Agile Design Methodology for Efficient, Reliable, and Secure ML Systems”, 2022 IEEE 40th VLSI Test Symposium (VTS), San Diego, CA, USA, 2022, pp. 1-14, doi: 10.1109/VTS52500.2021.9794253.

[7] J. Pasuksmit et al., “Improving Agile Planning for Reliable Software Delivery”, 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, Australia, 2023, pp. 25-26, doi: 10.1109/MSR59073.2023.00017.

[8] van Driel, W.D.; Bikker, J.W.; Tijink, M.; Di Bucchianico, A. Software Reliability for Agile Testing. Mathematics 2020, 8, 791. https://doi.org/10.3390/math8050791

[9] Soroush Aghamohammadi-Bosjin, Masoud Rabbani & Reza Tavakkoli-Moghaddam (2020) Agile two-stage lot-sizing and scheduling problem with reliability, customer satisfaction and behaviour under uncertainty: a hybrid metaheuristic algorithm, Engineering Optimization, 52:8, 1323-1343, DOI: 10.1080/0305215X.2019.1650923

[10] Barbareschi, M., Barone, S., Carbone, R. et al. Scrum for safety: an agile methodology for safety-critical software systems. Software Qual J 30, 1067–1088 (2022). https://doi.org/10.1007/s11219-022-09593-2

[11] G. Rathi, U. K. Tiwari and N. Singh, “Software Reliability: Elements, Approaches and Challenges”, 2022 International Conference on Advances in Computing, Communication and Materials (ICACCM), Dehradun, India, 2022, pp. 1-5, doi: 10.1109/ICACCM56405.2022.10009422.

[12] Lee, D.H.; Chang, I.H.; Pham, H. Software Reliability Growth Model with Dependent Failures and

Uncertain Operating Environments. Appl. Sci. 2022, 12, 12383. https://doi.org/10.3390/app122312383
[13] Md. Asraful Haque, Nesar Ahmad, Software reliability modeling under an uncertain testing environment, International Journal of Modelling and Simulation, 10.1080/02286203.2023.2201905, (1-7), (2023).
[14] Umashankar Samal & Ajay Kumar (2023) Redefining software reliability modeling: embracing faultdependency, imperfect removal, and maximum fault considerations, Quality Engineering, DOI: 10.1080/08982112.2023.2241067
[15] M. Nafreen, M. Luperon, L. Fiondella, V. Nagaraju, Y. Shi and T. Wandji, "Connecting Software Reliability Growth Models to Software Defect Tracking", 2020 IEEE 31st International Symposium on Software
[16] Reliability Engineering (ISSRE), Coimbra, Portugal, 2020, pp. 138-147, doi:
[17] 10.1109/ISSRE5003.2020.00022.
[18] K. K. Raghuvanshi, A. Agarwal, K. Jain, and V. B. Singh, "A time-variant fault detection software reliability model," SN Applied Sciences, vol. 3, no. 18, 2021.
[19] Kuldeep Singh Kaswan, Sunita Choudhary, Santar Pal Singh, Anil AudumbarPise, and Simon Karanja
[20] Hinga, "A New Variant of JM Software Reliability Model", Scientific Programming, Hindawi, Volume 2022, Article ID 7257564, 11 pages, https://doi.org/10.1155/2022/7257564
[21] Li, S.; Dohi, T.; Okamura, H. Are Infinite-Failure NHPP-Based Software Reliability Models Useful?
[22] Software 2023, 2, 1-18. https://doi.org/10.3390/software2010001

ETHICAL DECLARATION

Conflict of interest: No declaration required. **Financing:** No reporting required. **Peer review:** Double anonymous peer review.