Hindawi
Scientific Programming
Volume 2022, Article ID 7257564, 11 pages
https://doi.org/10.1155/2022/7257564

Research Article

A New Variant of JM Software Reliability Model

Kuldeep Singh Kaswan,[1] Sunita Choudhary,[2] Santar Pal Singh,[3] Anil Audumbar Pise,[4] and Simon Karanja Hinga[5]

[1]*School of Computing Science & Engineering, Galgotias University, Greater Noida-203201, India*
[2]*Department of Computing Science, Banasthali Vidyapith, Vanasthali, Rajasthan-304022, India*
[3]*Department of Computer Science & Engineering, Thapar Institute of Engineering and Technology, Patiala-147004, India*
[4]*School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg-2000, Gauteng, South Africa*
[5]*Department of Electrical and Electronic Engineering, Technical University of Mombasa, Mombasa, Kenya*

Correspondence should be addressed to Simon Karanja Hinga; kahinga@tum.ac.ke

Received 27 November 2021; Accepted 21 January 2022; Published 25 February 2022

Academic Editor: Punit Gupta

Software reliability is the probability of failure-free operations of software in a specific environment in a given time period. Various software reliability models have been designed by the researchers, but the JM model is the first influential model. The JM model was developed with the basic assumption that the faults are independent in this model and the debugging process is perfect. But practically, all debugging processes may not be perfect, especially when the faults are dependent; in this case, the fault that is actually to have been removed may also remove more than one fault and cause it to add some new faults. To handle this behavior of faults mutual dependency, we need a new model which may be less reliable or the result accuracy of the model may be lower than that of the existing ones, but it can handle more practical situations in the fault removal process. In this paper, we proposed a new software reliability model with the same assumption that at whatever time a failure is detected, it is not completely eradicated and there is a possibility of raising some new faults because of wrong analysis or inaccurate modifications in the software or the removal of the existing fault may also remove some other faults. The proposed model is more practical than the existing ones.

1. Introduction

Software reliability models are used to find the faults in a software product, and for the prediction of faults, these models predict and estimate the number of faults in the build. On behalf of this, one can take the decision whether this product has to be released or corresponding changes have to be made to improve the quality. Nowadays, due to the usage of software in real-time applications, even a single fault in the software becomes very critical, and it may result in the loss of life and other consequences. So, researchers are putting their best efforts in developing and improving the software reliability models so that it may help to provide more reliable software and better-quality software. Several software reliability models were proposed, but still the industry crept around the faults and unstable software. The JM model states that faults are independent of each other and equally likely to cause a failure during a test. The detected fault is eliminated immediately without the detection of any new fault. But these assumptions are not realistic. In our proposed work, we have extended the JM model by replacing these assumptions with the new assumptions that the faults are dependent and not equally likely to cause a failure in the test, and whenever a failure occurred, the identified faults are removed with probability p, and it may result in the removal and generation of some other faults, from the total number of faults, with random probabilities r, such that $p > r$, respectively.

This paper is organized into the following sections. Related work is given in Section 2. JM model, its assumptions, and the mathematical formulation are described in Section 3. In Section 4, we have proposed a new variant of the JM model. Results and discussions are given in Section 5. Finally, Section 6 concludes the work.

2. Related Work

The first reliability model was reported in 1967 using the Weibull distribution of time between failures [1]. After this, in 1972, the first influential software reliability model [2] with initial N bugs was reported [2]. A similar Jelinski–Moranda (JM) model was developed in 1975 [3, 4]. Some researchers designed the first nonhomogeneous Poisson process model [5]. In 1983, Meinhold and Singpurwalla proposed a Bayesian software reliability model which was a variant of the JM model that used the prior distributions to the parameters [6]. Jewell used the Meinhold and Singpurwalla model to derive a new model that provides Bayesian analysis of the software reliability model of JM [7]. Tohma proposed a new software reliability model [8] for estimating the number of residual software faults based on the hypergeometric distribution [8, 9]. Brocklehurst improved reliability predictions by a process of recalibration [10]. Sahinoglu uses the probability density estimation of failures in the clustering event of the software failures [11]. Campodonico and Singpurwalla proposed a Bayesian approach using the logarithmic Poisson model to predict the number of failures in a software system [12]. Chen and Arlat proposed a fault correction history-based input domain reliability growth model [13]. A software reliability model using enhanced nonhomogeneous Poisson process (ENHPP) approach was reported in the literature [14]. Some authors considered the phenomena of failure correlation to develop a software reliability model framework [15]. Tian described a model for homogeneous failure intensities by grouping data into clusters [16]. Huang estimated the reliability with the unified scheme of some nonhomogeneous Poisson process models [17]. Some researchers proposed a model for individual component-based software reliability and the architecture of the system [18]. Advanced chaos theory to the stochastic models, an alternative approach of software reliability, is also there in the literature [19].

Raj Kiran and Ravi group different models to accurately forecast software reliability [20]. Jun-Gang proposed an RVM (relevance vector machine)-based model for software reliability prediction [21]. Some researchers addressed the issue of optimal selection of software reliability growth models [22].

An improved additive model to reliability estimation of modular structure-based software is there to study [23]. Inoue and Yamada discussed discrete software reliability measurement based on a discredited (NHPP) model [24]. Kiyoshi Honda prosposed a stochastic process based software reliability model [25], and Kim HeeCheul proposed a comparative problem of a reliability model for Lomax and Gompertz distribution property [26]. Shinji Inoue proposed a new software reliability model with the effect of a change point for the Markovian software reliability model having an imperfect debugging environment [27]. This proposed model shows that the observed time-dependent behavior of the expected number of failures occurred after the change point has more practical situations compared to the other existing models. Kwang Yoon Song proposed a new nonhomogeneous Poisson process (NHPP) software reliability model [28]. An explicit mean value function solution for the proposed model is presented. Jinyong Wang and Xiaoping Mi proposed a new software reliability model [29] considering the decreasing trend of fault detection rate. This model has better predictive performance and better fitting than the previous existing models in this field. Yoshinobu Tamura and Shigeru Yamada proposed a deep learning-based scheme for the optimal selection of a software reliability model [30]. As model selection affects the optimal release time and total software cost, in this paper we also discussed these two criteria for the selection of a software reliability model. Subhashis Chatterjee and Ankur Shukla developed a new software reliability method with the imperfect debugging phenomenon [31]. A new ranking method has been proposed to improve the accuracy of model ranking. Da Hye Lee proposed a software reliability model based on NHPP [32]. The proposed model has the same mean value functions and the testing coverage, but it considers the environment that is uncertain. There are unexpected variables like syntax error considered in the proposed model. Shozab Khurshid designed a generalized framework to develop an effort-based software reliability model [33] with fault reduction factor (FRF), change point, and error generation. Yunlu Zhao, Tadashi Dohi, and Hiroyuki Okamura proposed a nonhomogeneous binomial processes (NHBPs)-based framework [34] for test-run reliability modeling. This paper also demonstrates that Poisson binomial distribution has a vital role in reliability modeling. Barack and Huang [35] proposed software reliability growth models (SRGMs) to assess and predict the reliability of a mobile application. Through the analysis of bug reports, four software reliability models are used to assess the dependability of an open-source mobile application. Sun and Li [36] proposed a new nonhomogeneous Poisson process (NHPP) based on fault severity considerations. We categorise software faults into three levels based on their complexity: Level I denotes a simple fault, Level II a general fault, and Level III a severe fault. Raghuvanshi et al. [37] proposed a time-variant software reliability model (SRM) that takes fault detection and the highest number of faults in software into account. The time-variant genetic algorithm process is used to evaluate the SRM parameters. The proposed model is based on a nonhomogeneous Poisson process (NHPP) and includes fault-dependent detection, software failure intensity, and unremoved error in the software. Van Driel et al. [38] predict the software reliability in agile testing environments and attempt to model this way of working by extending the Jelinski–Moranda model to a "stack" of feature-specific models, assuming that bugs are labelled with the feature to which they belong.

3. Jelinski–Moranda (JM) Model

The Jelinski–Moranda (JM) model [4] is a Markov model, and this model has strongly influenced many later models.

Numerous software reliability models have been proposed by assuming this model as the base model.

Characteristics of the JM model are as follows:

(1) It is a binomial type model
(2) It is probably the first and definitely one of the well-recognized black-box models
(3) This model always produces an overoptimistic reliability prediction
(4) JM model follows a perfect debugging process

3.1. Model Assumptions. The assumptions considered in the JM model are given as follows:

(i) There are unknown numbers of faults in the software initially and these fault counts are fixed and constant
(ii) The faults are not dependent on each other and equally likely to cause a failure during a test
(iii) There are independent time intervals among the occurrences of failures, exponentially distributed random variables
(iv) The software failure rate remains constant over the intervals between fault occurrences
(v) The failure rate is directly proportional to the number of faults that linger in the software
(vi) A detected fault is eliminated immediately, and no new faults are initiated during the elimination of the detected fault
(vii) When a failure occurs, the corresponding fault is removed with certainty

3.2. Mathematical Formulation of the JM Model

(i) Software fault rate: it is defined as the faults per unit time

$$\lambda(t_i) = \phi[N-(i-1)] \quad \text{where } i = 1, 2, \ldots, N, \tag{1}$$

in which ϕ is a constant of proportionality representing the failure rate contributed by each fault, N is the initial number of faults in the software, and t_i is the time between $(i-1)$ th and ith failure.

(ii) Failure density function: it is the function that assigns to each number the probability that the random variable takes a value less than or equal to the given number.

It is defined as the derivative of the failure probability.

$$f(t_i) = \phi[N-(i-1)]\exp(-\phi[N-(i-1)]t_i). \tag{2}$$

(iii) Distribution function is given as follows:

$$F_i(t_i) = 1 - \exp(\phi[N-(i-1)]t_i). \tag{3}$$

(iv) Reliability function at the ith failure interval is given by

$$R(t_i) = 1 - F_i(t_i) = \exp(-\phi[N-(i-1)]t_i). \tag{4}$$

(v) MTTF for the ith failure $= 1/\phi[N-(i-1)]$.

4. Proposed Model

The assumptions (ii) and (vi) of the JM model states that faults are independent of each other and equally likely to cause a failure at some point in a test. The detected fault is removed immediately without the detection of any new fault. But these assumptions are not realistic. We extended the JM model by replacing the assumptions (ii) and (vi) with the new assumptions that the faults are dependent and not equally likely to cause a failure during a test, and whenever a failure occurred, the detected faults are eliminated with probability p, and it may result in the removal and generation of some other faults, from the total number of faults, with random probabilities r, such that $p > r$, respectively.

4.1. Model Assumptions of the Proposed Model. The assumptions in the proposed model include the following:

(i) to (v) Assumptions (i) to (v) are the same as of the JM model

vi) Whenever a failure occurred, the detected faults are removed with some probability and it may result first in the removal of some other faults with the random probability p and second in the generation of some other new faults with the random probability r, such that $p > r$.

4.2. Mathematical Formulation of the Proposed Model

(i) Failure rate:

$$\lambda(t_i) = \Phi\left[N-(i-1)\left\{\frac{\sum_{j=i}^{N} pj}{N-(i-1)} - \frac{\sum_{k=1}^{m} rk}{m}\right\}\right], \tag{5}$$

where φ is the proportionality constant representing the failure rate contributed by each fault, N is the initial no. of faults in the software, t_i is the time between $(i-1)$th and ith failure, p_j is the random probability to remove the faults, r_k is the random probability to add some new faults, and m is the number of faults added such that $p_j > r_k$ and $m < N-(i-1)$.

(ii) Failure density is defined as "at any point in the life of a system, the incremental change in the number of failures per associated incremental change in time"

$$f\left(t_i\right)=\Phi\left[N-(i-1)\left\{\frac{\sum_{j=i}^{N} pj}{N-(i-1)}-\frac{\sum_{k=1}^{m} rk}{m}\right\}\right]\exp\left[-\Phi\left(N-(i-1)\left\{\frac{\sum_{j=i}^{N} pj}{N-(i-1)}-\frac{\sum_{k=1}^{m} rk}{m}\right\}\right)t_i\right]. \tag{6}$$

The failure distribution function is the integral of the failure density function.

(iii) Distribution function (cumulative density function):

$$F_i\left(t_i\right)=1-\exp\left[-\Phi\left(N-(i-1)\left\{\frac{\sum_{j=i}^{N} pj}{N-(i-1)}-\frac{\sum_{k=1}^{m} rk}{m}\right\}\right)t_i\right] \tag{7}$$

or

$$F_i\left(t_i\right)=1-\exp\left[-\lambda_i t_i\right]. \tag{8}$$

(iv) The mean time to failure (MTTF) is the average time between observed failures: MTTF $= 1 - F_i(t_i)$.

(v) Reliability function:

$$R\left(t_i\right)=1-F_i\left(t_i\right)=\exp\left[-\Phi\left(N-(i-1)\left\{\frac{\sum_{j=i}^{N} pj}{N-(i-1)}-\frac{\sum_{k=1}^{m} rk}{m}\right\}\right)t_i\right]. \tag{9}$$

4.3. Parameter Estimation. We have to estimate the number of remaining faults N' and the constant of proportionality Φ. Our proposed model parameters are estimated using the maximum likelihood estimation method.

(i) Parameter estimation:

$$\begin{aligned}\sum_{i=1}^{n}&\frac{1}{N'-(i-1)\left[\sum_{j=i}^{N} pj/N-(i-1)-\sum_{k=1}^{m} rk/m\right]}\\&=\frac{n}{N'-\left(1/\sum_{i=1}^{n} tn\right)\left[\sum_{i=1}^{n}(i-1)\left\{\sum_{j=i}^{N} pj/N-(i-1)-\sum_{k=1}^{m} rk/m\right\}t_i\right]},\end{aligned} \tag{10}$$

$$\Phi=\frac{n}{\sum_{i=1}^{n} N'-(i-1)\left[\sum_{j=i}^{N} pj/N-(i-1)-\sum_{k=1}^{m} rm/m\right]t_i}. \tag{11}$$

We have obtained maximum likelihood estimation N' by solving the equation (10) and put this value into (11) to obtain the maximum likelihood estimation Φ.

A program has been implemented in MATLAB to find the value of N' from (5) (Algorithm 1)

$$\begin{aligned}f\left(N'\right)=\sum_{i=1}^{n}&\frac{1}{N'-(i-1)\left[\sum_{j=i}^{N} pj/N-(i-1)-\sum_{k=1}^{m} rk/m\right]}\\&-\frac{n}{N'-\left(1/\sum_{i=1}^{n} tn\right)\left[\sum_{i=1}^{n}(i-1)\left\{\sum_{j=i}^{N} pj/N-(i-1)-\sum_{k=1}^{m} rk/m\right\}t_i\right]}.\end{aligned} \tag{12}$$

Now find the reliability for the next time interval.

(i) Reliability:

(1) for $n = 3$ to 136
  begin
(2) for $N' = 3$ to 150
   begin
(3) $r = f(N')$
   end
(4) Find the minimum value of r and print N' for that value.
  end

Algorithm 1: Algorithm to estimate N' from equation (5).

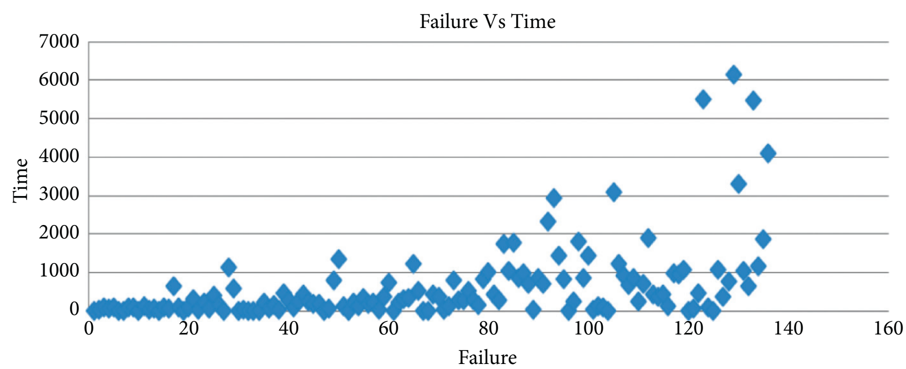

Figure 1: Failure vs. time.

Table 1: Proposed model vs. JM model.

N	t_n	JM model					Proposed model				
		N'	ϕ	λ	MTTF	$R = 1 - f(t)$	N'	ϕ	λ	MTTF	$R = 1 - f(t)$
1	3	1	0.333300	0.000000	∞	1.000000	1	0.333333	0.333333	3	0.367879
2	30	2	0.055600	0.000000	∞	1.000000	2	0.036263	0.059414	16.831049	0.168230
3	113	3	0.016500	0.000000	∞	1.000000	3	0.009324	0.018380	54.406556	0.125311
4	81	4	0.009800	0.000000	∞	1.000000	4	0.004840	0.014483	69.0432864	0.309382
5	115	6	0.004600	0.004600	218.600000	0.959700	6	0.002671	0.011844	84.430767	0.256131
6	9	11	0.002100	0.010500	95.233300	0.979200	6	0.002871	0.011980	83.469009	0.897785
7	2	∞	0.000000	0.019800	50.428600	0.164600	7	0.002838	0.013825	72.332568	0.972728
8	91	28	0.000742	0.014800	67.368800	0.189700	8	0.002414	0.012989	76.983480	0.306642
9	112	16	0.001400	0.009900	100.746000	0.861700	11	0.001567	0.012004	83.300114	0.260660
10	15	46	0.000424	0.015300	65.505600	0.121600	12	0.001471	0.012387	80.726542	0.830429
11	138	20	0.001100	0.009800	102.181800	0.613000	NaN	NaN	NaN	NaN	NaN
12	50	27	0.000756	0.011300	88.216700	0.417800	12	0.001353	0.009844	101.574951	0.611251
13	77	29	0.000695	0.011100	89.932700	0.765800	13	0.001233	0.010802	92.573668	0.435278
14	24	61	0.000300	0.014100	70.835900	0.217700	14	0.001175	0.010457	95.6241495	0.778035
15	108	39	0.000494	0.011800	84.416700	0.352600	15	0.001079	0.009997	100.022048	0.339676
16	88	38	0.000509	0.011200	89.335200	0.000600	16	0.000978	0.009748	102.584017	0.424079
17	670	18	0.001400	0.001500	686.235300	0.839600	17	0.000665	0.007561	132.245087	0.006305
18	120	20	0.001200	0.002300	429.944400	0.941300	18	0.000554	0.006507	153.677753	0.458014
19	26	23	0.000899	0.003600	278.236800	0.663800	22	0.000463	0.007150	139.855327	0.830351
20	114	25	0.000782	0.003900	255.740000	0.280600	20	0.000515	0.006244	160.132492	0.490705
21	325	24	0.000844	0.002500	395.047600	0.870000	21	0.000458	0.005896	169.592304	0.147141
22	55	27	0.000684	0.003400	292.281800	0.436900	23	0.000408	0.005975	167.343635	0.719884
23	242	28	0.000639	0.003200	312.773900	0.804600	25	0.000363	0.006200	161.274074	0.223007
24	68	31	0.000541	0.003800	263.910700	0.202100	24	0.000377	0.005658	176.711764	0.680581
25	422	29	0.000608	0.002400	410.950000	0.645300	30	0.000281	0.005879	170.070077	0.083631
26	180	31	0.000538	0.002700	372.084600	0.973500	28	0.000290	0.004883	204.750312	0.415148
27	10	35	0.000439	0.003500	285.060200	0.017900	27	0.000304	0.005190	192.65184	0.949417

Table 1: Continued.

N	t_n	JM model					Proposed model				
		N'	ϕ	λ	MTTF	$R = 1 - f(t)$	N'	ϕ	λ	MTTF	$R = 1 - f(t)$
28	1146	30	0.000576	0.001200	867.339300	0.500700	28	0.000249	0.004452	224.595888	0.006081
29	600	31	0.000529	0.001100	944.913800	0.984300	29	0.000208	0.003817	261.922210	0.101189
30	15	33	0.000462	0.001400	721.477800	0.951300	34	0.000175	0.003886	257.306309	0.943370
31	36	35	0.000412	0.001600	606.540300	0.993400	31	0.000197	0.004052	246.749856	0.864246
32	4	38	0.000354	0.002100	471.322900	1.000000	35	0.000180	0.004712	212.192154	0.981325
33	0	41	0.000312	0.002500	400.609800	0.980200	39	0.000166	0.004508	221.801057	1
34	8	46	0.000259	0.003100	321.838200	0.493900	34	0.000196	0.004231	236.344719	0.966717
35	227	47	0.000251	0.003000	331.804800	0.822100	40	0.000167	0.004566	219.000994	0.354685
36	65	52	0.000215	0.003400	290.074700	0.545100	36	0.000186	0.004050	246.873528	0.768516
37	176	54	0.000204	0.003500	287.804500	0.817500	37	0.000182	0.004212	237.414285	0.476484
38	58	60	0.000176	0.003900	258.077800	0.170200	44	0.000154	0.004847	206.272099	0.754890
39	457	55	0.000200	0.003200	313.152200	0.383700	46	0.000143	0.004333	230.744445	0.137993
40	300	56	0.000194	0.003100	322.792200	0.740400	NaN	NaN	NaN	NaN	NaN
41	97	60	0.000175	0.003300	300.445400	0.416700	41	0.000155	0.004072	245.535290	0.673642
42	263	61	0.000171	0.003200	308.000000	0.230500	51	0.000124	0.004525	220.977867	0.304171
43	452	58	0.000185	0.002800	360.924000	0.493400	53	0.000155	0.004254	235.032626	0.146147
44	255	60	0.000175	0.002800	357.265600	0.576100	52	0.000155	0.004091	244.438673	0.352323
45	197	62	0.000167	0.002800	352.882400	0.578700	46	0.000129	0.003872	258.217142	0.466301
46	193	65	0.000155	0.002900	339.527500	0.982500	47	0.000126	0.003811	262.382299	0.479233
47	6	71	0.000137	0.003300	304.892700	0.771700	50	0.000120	0.004016	248.944067	0.976186
48	79	77	0.000122	0.003500	282.576900	0.055700	48	0.000127	0.003553	281.394045	0.755220
49	816	67	0.000149	0.002700	373.731300	0.026900	49	0.000118	0.003763	265.736415	0.046388
50	1351	59	0.000183	0.001600	607.193300	0.783700	52	0.000100	0.003386	295.265642	0.010300
51	148	61	0.000173	0.001700	578.515700	0.964400	51	0.000098	0.003147	317.685879	0.627589
52	21	65	0.000155	0.002000	497.463000	0.626000	56	0.000091	0.003221	310.373974	0.934577
53	233	67	0.000147	0.002100	485.574100	0.758800	53	0.000096	0.003241	308.498330	0.469883
54	134	70	0.000137	0.002200	456.072900	0.457100	54	0.000095	0.003194	312.990824	0.651728
55	357	71	0.000134	0.002100	466.751100	0.661300	NaN	NaN	NaN	NaN	NaN
56	193	74	0.000125	0.002300	443.803600	0.587600	56	0.000090	0.003282	304.614060	0.530683
57	236	76	0.000120	0.002300	438.064600	0.931700	NaN	NaN	NaN	NaN	NaN
58	31	81	0.000109	0.002500	398.967800	0.396600	58	0.000087	0.003455	289.420777	0.898426
59	369	81	0.000109	0.002400	416.571600	0.166000	70	0.000072	0.003446	290.145967	0.280333
60	748	78	0.000116	0.002100	481.008300	1.000000	60	0.000082	0.002949	339.070155	0.110136
61	0	82	0.000107	0.002200	444.750200	0.593500	72	0.000067	0.003388	295.134138	1
62	232	85	0.000101	0.002300	429.852700	0.464100	62	0.000079	0.002638	379.051733	0.542236
63	330	86	0.000099	0.002300	437.323000	0.434000	69	0.000070	0.003359	297.657092	0.330001
64	365	87	0.000098	0.002200	445.354600	0.064300	65	0.000074	0.003250	307.669056	0.305336
65	1222	81	0.000109	0.001700	572.519200	0.387300	65	0.000070	0.002786	358.894706	0.033210
66	543	81	0.000109	0.001600	610.210100	0.983700	66	0.000066	0.002768	361.168940	0.222361
67	10	85	0.000101	0.001800	551.660000	0.971400	67	0.000066	0.002500	399.861661	0.975301
68	16	89	0.000094	0.002000	508.892200	0.353600	68	0.000065	0.002919	342.534350	0.954363
69	429	89	0.000094	0.001900	534.642800	0.492200	69	0.000064	0.002675	373.720337	0.242804
70	379	90	0.000092	0.001800	543.980000	0.922300	70	0.000062	0.002393	417.854376	0.403727
71	44	94	0.000086	0.002000	506.655200	0.775200	71	0.000062	0.002759	362.371404	0.885659
72	129	98	0.000080	0.002100	478.508000	0.184000	NaN	NaN	NaN	NaN	NaN
73	810	95	0.000084	0.001900	538.806400	0.583800	73	0.000059	0.002668	374.698345	0.115124
74	290	97	0.000082	0.001900	532.678600	0.569400	74	0.000058	0.002778	359.874367	0.446713
75	300	99	0.000079	0.001900	527.241700	0.366700	75	0.000057	0.002715	368.299667	0.442836
76	529	99	0.000079	0.001800	550.189400	0.600100	76	0.000055	0.002621	381.517830	0.249931
77	281	101	0.000077	0.001800	544.009700	0.745200	77	0.000054	0.002564	389.939847	0.486448
78	160	105	0.000072	0.001900	514.758800	0.200200	78	0.000054	0.002684	372.443989	0.650772
79	828	102	0.000075	0.001700	576.648300	0.173200	79	0.000052	0.002524	396.126421	0.123657
80	1011	99	0.000079	0.001500	664.027600	0.511600	80	0.000050	0.002304	433.903508	0.097294
81	445	101	0.000076	0.001500	654.198100	0.636100	81	0.000048	0.002689	371.789765	0.302125
82	296	103	0.000074	0.001600	643.633600	0.065400	82	0.000047	0.002356	424.331389	0.497794
83	1755	98	0.000081	0.001200	827.210400	0.276300	83	0.000045	0.002230	448.354358	0.019954
84	1064	97	0.000082	0.001100	935.631900	0.148700	NaN	NaN	NaN	NaN	NaN
85	1783	95	0.000086	0.000856	1168.300000	0.479000	85	0.000040	0.001948	513.315339	0.031008
86	860	96	0.000084	0.000836	1195.900000	0.439600	NaN	NaN	NaN	NaN	NaN

Table 1: Continued.

N	t_n	JM model					Proposed model				
		N'	ϕ	λ	MTTF	$R=1-f(t)$	N'	ϕ	λ	MTTF	$R=1-f(t)$
87	983	96	0.000084	0.000754	1326.000000	0.586700	NaN	NaN	NaN	NaN	NaN
88	707	97	0.000082	0.000738	1354.900000	0.975900	88	0.000035	0.001920	520.822995	0.257311
89	33	100	0.000077	0.000845	1183.700000	0.480300	92	0.000034	0.001999	500.222704	0.936158
90	868	100	0.000077	0.000770	1298.200000	0.572500	94	0.000033	0.002061	485.139850	0.167098
91	724	101	0.000075	0.000755	1325.000000	0.173200	NaN	NaN	NaN	NaN	NaN
92	2323	100	0.000077	0.000615	1625.800000	0.164900	99	0.000029	0.001955	511.391098	0.010646
93	2930	99	0.000079	0.000471	2123.100000	0.502500	105	0.000026	0.001861	537.187177	0.004277
94	1461	99	0.000079	0.000394	2539.200000	0.717500	94	0.000028	0.001583	631.710908	0.098986
95	843	101	0.000075	0.000448	2233.200000	0.994600	104	0.000024	0.001666	599.918456	0.245320
96	12	102	0.000073	0.000439	2275.400000	0.891600	96	0.000027	0.001485	673.395806	0.982337
97	261	104	0.000070	0.000489	2044.300000	0.414600	116	0.000022	0.001836	544.648991	0.619273
98	1800	104	0.000070	0.000420	2382.100000	0.695500	121	0.000021	0.001641	609.064039	0.052060
99	865	106	0.000067	0.000466	2145.700000	0.512300	99	0.000021	0.001656	603.577182	0.238562
100	1435	106	0.000067	0.000401	2495.100000	0.988000	NaN	NaN	NaN	NaN	NaN
101	30	108	0.000064	0.000447	2236.700000	0.938100	116	0.000021	0.001310	762.984731	0.961443
102	143	110	0.000061	0.000490	2042.500000	0.948500	110	0.000022	0.001477	677.017482	0.809595
103	108	112	0.000059	0.000529	1890.100000	1.000000	103	0.000024	0.001442	693.326793	0.855755
104	0	114	0.000057	0.000566	1766.100000	0.171900	134	0.000018	0.001799	555.814849	1
105	3110	113	0.000058	0.000461	2169.600000	0.562800	105	0.000023	0.001441	693.594304	0.011289
106	1247	114	0.000056	0.000451	2215.900000	0.653400	106	0.000022	0.001402	712.758614	0.173853
107	943	115	0.000055	0.000443	2259.600000	0.733600	131	0.000017	0.001575	634.876873	0.226428
108	700	116	0.000054	0.000435	2301.000000	0.683700	108	0.000021	0.001388	720.189902	0.378338
109	875	118	0.000052	0.000469	2134.000000	0.891500	109	0.000020	0.001355	737.558886	0.305334
110	245	119	0.000051	0.000462	2166.700000	0.714300	110	0.000020	0.001413	707.414134	0.707277
111	729	121	0.000049	0.000493	2028.800000	0.392600	121	0.000018	0.001203	830.853083	0.415857
112	1897	121	0.000049	0.000444	2252.900000	0.820000	112	0.000019	0.001380	724.616992	0.072953
113	447	123	0.000047	0.000475	2106.100000	0.832500	118	0.000018	0.001553	643.567841	0.499291
114	386	124	0.000047	0.000468	2137.400000	0.811700	NaN	NaN	NaN	NaN	NaN
115	446	126	0.000045	0.000497	2014.000000	0.941200	115	0.000019	0.001209	826.530574	0.582978
116	122	128	0.000044	0.000524	1908.000000	0.595200	116	0.000019	0.001299	769.428084	0.853372
117	990	129	0.000043	0.000516	1938.900000	0.613300	117	0.000018	0.001422	702.849012	0.244496
118	948	131	0.000041	0.000539	1854.200000	0.557900	118	0.000019	0.001006	993.772526	0.385220
119	1082	132	0.000041	0.000531	1884.200000	0.988400	119	0.000018	0.001129	885.207957	0.294548
120	22	134	0.000040	0.000555	1802.500000	0.959200	120	0.000020	0.001302	767.619405	0.971746
121	75	136	0.000039	0.000578	1731.300000	0.757000	121	0.000018	0.000978	1021.98930	0.929241
122	482	138	0.000037	0.000598	1672.000000	0.037100	122	0.000018	0.001441	693.576499	0.499100
123	5509	134	0.000040	0.000436	2292.400000	0.957300	123	0.000017	0.001153	866.636866	0.001734
124	100	136	0.000038	0.000461	2169.400000	0.995400	124	0.000016	0.001434	697.121298	0.866366
125	10	138	0.000037	0.000485	2063.700000	0.595100	125	0.000016	0.000697	1433.87795	0.993050
126	1071	139	0.000037	0.000477	2094.700000	0.837700	126	0.000016	0.001357	736.891254	0.233773
127	371	141	0.000036	0.000499	2004.600000	0.674300	127	0.000016	0.001108	902.406476	0.662905
128	790	143	0.000035	0.000518	1929.700000	0.041300	NaN	NaN	NaN	NaN	NaN
129	6150	139	0.000037	0.000367	2723.300000	0.295400	129	0.000015	0.000894	1118.172980	0.004086
130	3321	139	0.000037	0.000330	3031.000000	0.708400	130	0.000014	0.001045	956.790792	0.031086
131	1045	140	0.000036	0.000325	3079.800000	0.810300	159	0.000011	0.001370	729.439350	0.238685
132	648	141	0.000036	0.000320	3125.400000	0.172900	NaN	0.000008	NaN	NaN	NaN
133	5485	140	0.000036	0.000253	3953.600000	0.745700	206	0.000004	0.001080	925.678922	0.002670
134	1160	141	0.000036	0.000249	4020.900000	0.629000	373	0.000004	0.001466	681.741222	0.182405
135	1864	142	0.000035	0.000244	4094.400000	0.365900	NaN	NaN	NaN	NaN	NaN
136	4116	142	0.000035	0.000209	4777.000000	0.000000	NaN	NaN	NaN	NaN	NaN

$$R(tn+1) = 1 - Fn + 1(tn+1)$$
$$= \exp\left(-\Phi\left[(N-n)\left\{\frac{\sum_{j=i}^{N} pj}{N-n} - \frac{\sum_{k=1}^{m} rk}{m}\right\}\right]tn+1\right). \tag{13}$$

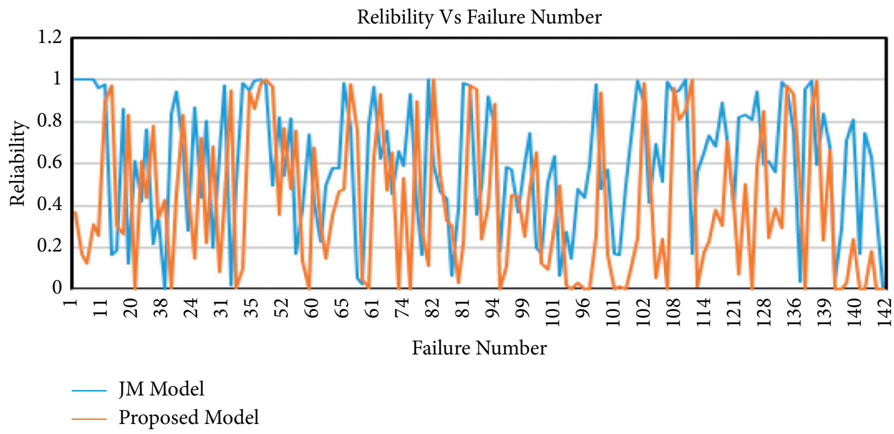

Figure 2: Reliability vs. failure number.

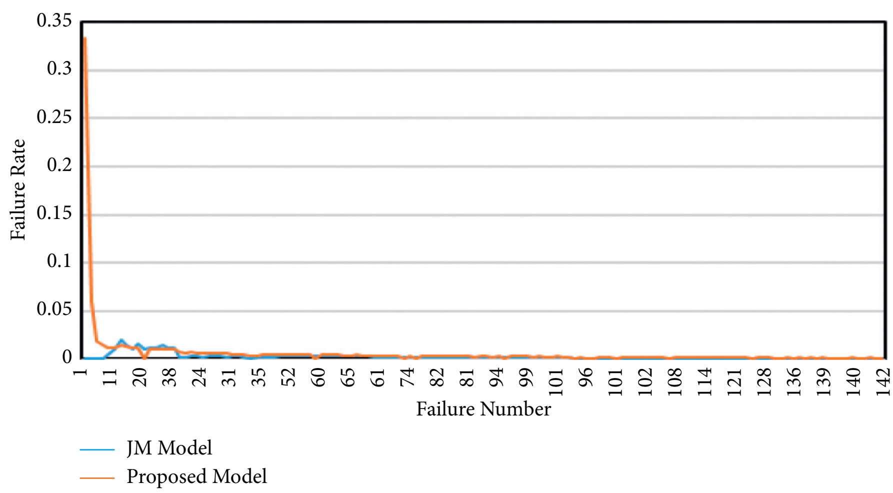

Figure 3: Failure rate vs. failure number.

5. Results and Discussion

In this paper, we proposed and implemented a new variant of the JM model. The failure vs. time graph based on the dataset used by Musa [9] is shown in Figure 1.

We estimate the parameters (ϕ, λ). With the help of these parameters, we calculate the mean time to failure (MTTF) and the reliability for the JM model and the proposed model using MATLAB R 2015a.

The model validation is given in Table 1.

It is concluded from this table that the reliability of the proposed model is not as expected as the JM model. But the proposed model assumptions are more realistic and will act as a new approach for software reliability estimation.

A response graph has been used to show the effect of individual input failure parameters on selected responses. The effect of the following one factor graphs (Figures 2–4) was studied on output.

(a) Reliability vs. failure number

(b) Failure rate vs. failure number

(c) Failure rate vs. MTTF

In Figure 2, we have compared the software reliability with the failure numbers. The result shows that the proposed model exhibits almost similar behavior as the JM model, and the proposed model is found to be more practical than the JM model. Figure 3 shows that the failure rate for the proposed model is greater than that for the JM model, as the proposed model is for imperfect debugging. Figure 4 compares MTTF and failure rate, and at some point, MTTF for the proposed model is less than that for the JM model.

We have compared the proposed model and the JM model in terms of average MTTF and the average reliability of the system and the results are shown in Figures 5–6.

From Figure 5, it has been found that the average mean time to failure (MTTF) for the proposed model is less than that for the JM model. This shows that, using the proposed model, we have improved the system.

From Figure 6, it has been found that the average reliability for the JM model is more than that for the proposed model, but the proposed model is more practical and has a better real-world approach.

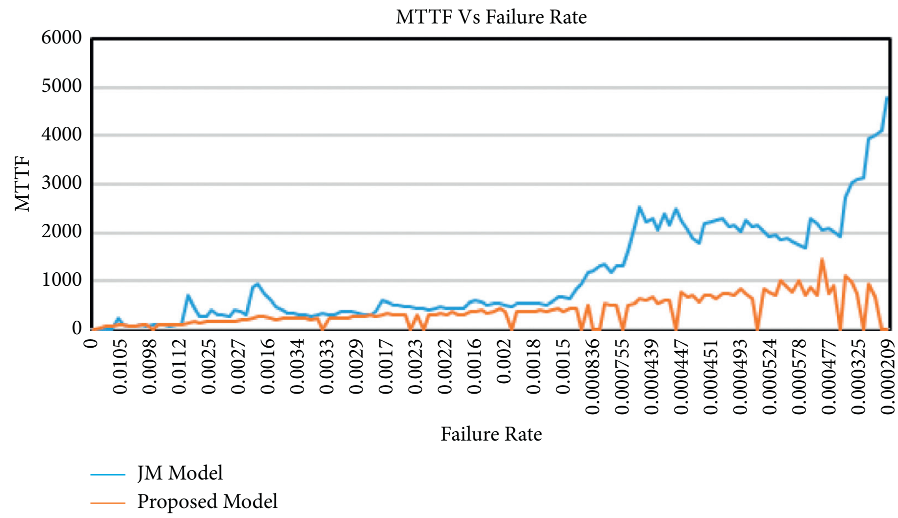

Figure 4: MTTF vs. failure rate.

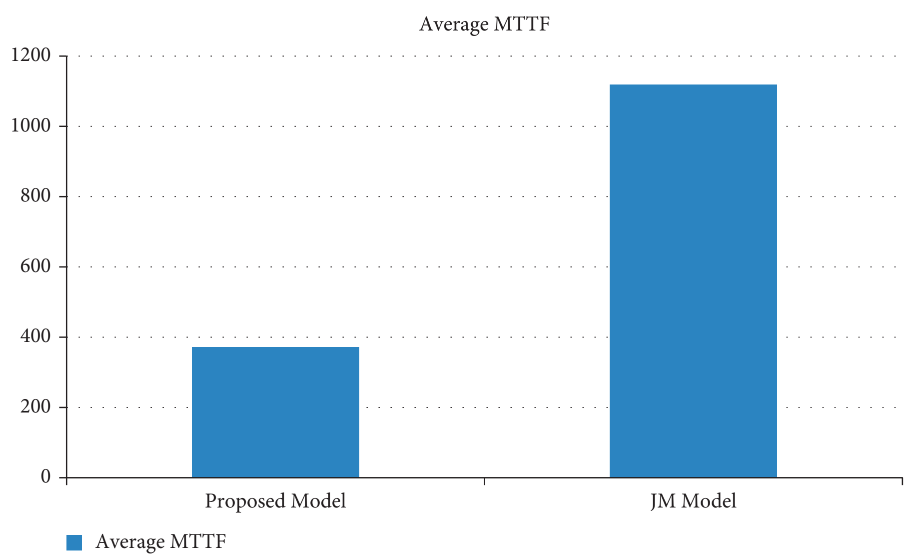

Figure 5: Comparison of the proposed model MTTF with the JM model MTTF.

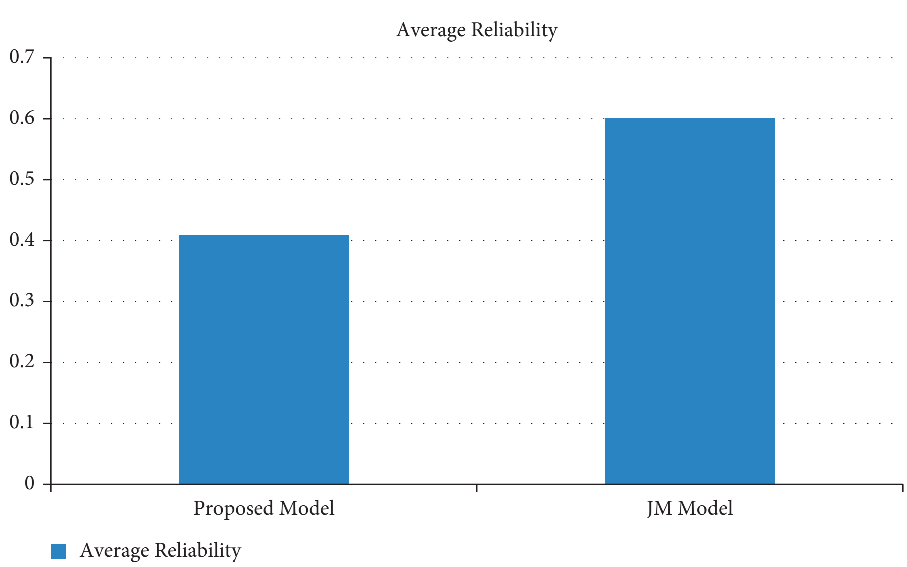

Figure 6: Comparison of the average reliability of the proposed model with the JM model reliability.

6. Conclusion

The proposed model is an imperfect debugging process model with fault dependency. In this model, the removal of the existing fault can also remove some other faults with the random probability of an individual and it may also generate some new faults with some probability. Reliability for the JM model and the proposed model is 0.6 and 0.4, respectively. The mean time to failure (MTTF) for the JM model and the proposed model is 1118.596 and 371.5370972, respectively. Experimental results indicate that MTTF for the proposed model is found to be better than that for the JM model. But the reliability of the proposed model is not as good as the JM model, but it has a more real-world approach and practical nature.

Data Availability

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of Interest

The authors declare that they do not have any conflicts of interest.

References

[1] G. R. Hudson, "Program error as a birth and death process," ReportNo. SP-3011, System Development Corporation, Santa Monica, CA, USA, 1967.

[2] Z. Jelinski and P. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation*, W. Freiberger, Ed., Academic Press, New York, NY, USA, pp. 465–484, 1972.

[3] M. L. Shooman, "Software reliability: measurement and models," in *Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 485–491, Washington, DC, USA, January 1975.

[4] J. D. Musa, "A theory of software reliability and its application," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 3, pp. 312–327, 1975.

[5] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 206–211, 1979.

[6] R. J. Meinhold and N. D. Singpurwalla, "Bayesian analysis of a commonly used model for describing software failures," *The Statistician*, vol. 32, no. 1/2, pp. 168–173, 1983.

[7] W. S. Jewell, "Bayesian extensions to a basic model of software reliability," *IEEE Transactions on Software Engineering*, vol. SE-I1, no. 12, pp. 1081–1091, 1985.

[8] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution," *IEEE Transactions on Software Engineering*, vol. 15, no. 3, pp. 345–355, 1989.

[9] J. D. Musa, "Data," Data & Analysis Center for Software, 1980, http://www.dacs.dtic.mil/databases/sledfswrel.shtml.

[10] S. Brocklehurst, P. Y. Chan, B. Littlewood, and J. Snell, "Recalibrating software reliability models," *IEEE Transactions on Software Engineering*, vol. 16, no. 4, pp. 458–470, 1990.

[11] M. Sahinoglu, "Compound-Poisson software reliability model," *IEEE Transactions on Software Engineering*, vol. 18, no. 7, pp. 624–630, 1992.

[12] S. Campodonico and N. D. Singpurwalla, "A bayesian analysis of logarithmic-Poisson execution time model based on expert opinion and failure data," *IEEE Transactions on Software Engineering*, vol. 20, no. 9, pp. 677–683, 1994.

[13] Y. Chen and J. Arla, "An Input Domain-Based Reliability Growth Model and its Applications in Comparing Software Testing Strategies," LAAS Report No.95105, LAAS-CNRS, France, 1995.

[14] S. S. Gokhale and K. S. Trivedi, "A time/structure-based software reliability model," *Annals of Software Engineering*, vol. 8, no. 1/4, pp. 85–121, 1999.

[15] K. G. Popstojanovaand, K. S. Trivedi, Failure correlation in software reliability models," *IEEE Transactions on Reliability*, vol. 49, no. 1, pp. 37–48, 2000.

[16] J. Tian, "Better reliability assessment and prediction through data clustering," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 997–1007, 2002.

[17] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A unified scheme of some nonhomogenous Poisson process models for software reliability estimation," *IEEE Transactions on Software Engineering*, vol. 29, no. 3, pp. 261–269, 2003.

[18] J.-H. Lo, C.-Y. Huang, I.-Y. Chen, S.-Y. Kuo, and M. R. Lyu, "Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure," *Journal of Systems and Software*, vol. 76, no. 1, pp. 3–13, 2005.

[19] S. Dick, C. L. Bethel, and A. Kandel, "Software-reliability modeling: the case for deterministic behavior," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 1, pp. 106–119, 2007.

[20] N. Raj Kiran and V. Ravi, "Software reliability prediction by soft computing techniques," *Journal of Systems and Software*, vol. 81, no. 4, pp. 576–658, 2007.

[21] J. G. Lou, J.-H. Jiang, C. Y. Shuai, R. Zhang, and A. Jin, "Software reliability prediction model based on relevance vector machine," in *Proceedings of the IEEE Int. Conf. on Intelligent Computing and Intelligent Systems*, pp. 229–233, Shanghai, China, November 2009.

[22] K. Sharma, R. Garg, C. K. Nagpal, and R. K. Garg, "Selection of optimal software reliability growth models using a distance based approach," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 266–276, 2010.

[23] S. Chatterjee, S. Nigam, J. B. Singh, and L. N. Upadhyaya, "An improved additive model for reliability analysis of software ,with modular structure," *J. Appl. Math. Informatics*, vol. 30, pp. 489–498, 2012.

[24] S. Inoue and S. Yamada, "A bootstrapping approach for software reliability measurement based on a discretized NHPP model," *Journal of Software Engineering and Applications*, vol. 6, no. 4, pp. 1–7, 2013.

[25] K. Honda, H. Washizaki, and Y. Fukazawa, "A generalized software reliability model considering uncertainty and dynamics in development," in *Proceedings of the Int. Conf. on Product Focused Software Process Improvement*, pp. 342–346, Trondheim, Norway, November 2016.

[26] H. C. Kim, "A performance analysis of software reliability model using Lomax and Gompertz distribution property," *Indian Journal of Scienec and Technology*, vol. 9, no. 20, pp. 1–6, 2016.

[27] S. Inoue and S. Yamada, "Software reliability modeling with imperfect debugging and change of test environment," in

Proceedings of the Sixth International Conference on Reliability, Infocom Technologies and Optimization Trends and Future Directions, pp. 128–131, ICRITO), Noida, India, September 2017.

[28] K. Y. Song, I. H. Chang, and H. Pham, "An NHPP software reliability model with S-shaped growth curve subject to random operating environments and optimal release time," appl," vol. 7, no. 12, Article ID 1304, 2017.

[29] J. Wang and X. Mi, "Open-source software reliability model with the decreasing trend of fault detection rate," *The Computer Journal*, vol. 62, no. 19, pp. 1301–1312, 2019.

[30] Y. Tamura and S. Yamada, "Software reliability model selection based on deep learning with application to the optimal release problem," *Journal of Industrial Engineering and Management Science*, vol. 2016, no. 1, pp. 43–58, Article ID 3, 2016.

[31] S. Chatterjee and A. Shukla, "A unified approach of testing coverage-based software reliability growth modeling with fault detection probability, imperfect debugging, and change point," *Journal of Software: Evolution and Process*, vol. 30, 2018.

[32] D. H. Lee, I. H. Chan, H. Pham, and K. Y. Song, "A software reliability model considering the syntax error in uncertainty environment, optimal release time, and sensitivity analysis," *Applied Science*, vol. 8, Article ID 1483, 2018.

[33] S. Khurshid, A. K. Shrivastava, and J. Iqbal, "Effort based software reliability model with fault reduction factor, change point and imperfect debugging," *International Journal of Information Technology*, vol. 13, 2019.

[34] Y. Zhao, T. Dohi, and H. Okamura, "Software test-run reliability modeling with non-homogeneous binomial processes," in *Proceedings of the 2018 IEEE Twenty Third Pacific Rim International Symposium on Dependable Computing*, pp. 145–154, PRDC), Taipei, Taiwan, December 2018.

[35] O. Barack and L. Huang, "Assessment and prediction of software reliability in mobile applications," *Journal of Software Engineering and Applications*, vol. 13, no. 9, pp. 179–190, 2020.

[36] X. Sun and J. Li, "Simulation of software reliability growth model based on fault severity and imperfect debugging," in *Proceedings of the Simulation Tools and Techniques, Twelveth EAI International Conference*, SIMUtools, Guiyang, China, August 2020.

[37] 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.

[38] W. D. Van Driel, J. W. Bikker, and M. Tijink, "Prediction of software reliability," *Microelectronics Reliability*, vol. 119, Article ID 114074, 2021.