New failure rate model for iterative software development life cycle process

Sangeeta¹,² · Sitender³ · Kapil Sharma⁴ · Manju Bala⁵

Received: 10 December 2020 / Accepted: 24 June 2021 / Published online: 9 July 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Software reliability models are one of the most generally used mathematical tool for estimation of reliability, failure rate and number of remaining faults in the software. Existing software reliability models are designed to follow waterfall software development life cycle process. These existing models do not take advantage of iterative software development process. In this paper, a new failure rate model centered on iterative software development life cycle process has been developed. It aims to integrate a new modulation factor for incorporating varying needs in each phase of iterative software development process. It comprises imperfect debugging with the possibility of fault introduction and removal of multiple faults in an interval as iterative development of the software proceeds. The proposed model has been validated on twelve iterations of Eclipse software failure dataset and nine iterations of Java Development toolkit (JDT) software failure dataset. Parameter estimation for the proposed model has been done by hybrid particle swarm optimization and gravitational search algorithm. Experimental results in-terms of goodness-of-fit shows that proposed model has outperformed Jelinski Moranda, Shick Wolverton, Goel Okumoto Imperfect debugging, GS Mahapatra, Modified Shick Wolverton in 83.33% of iterations for eclipse dataset and 77.77% of iterations for JDT dataset.

Keywords Software development life cycle · Iterative software development life cycle · Optimization · Failure rate model · Software reliability

1 Introduction

With the growing advances in the digital world, software development demand from industries is growing at an exponential rate. Due to enormous demand, lack of time and budget, software companies are not able to develop fault-free software.

* Sitender
sitender@msit.in

Extended author information available on the last page of the article
Latest tools and techniques have been applied for the development of defect-free software, but still, it is not possible for software developers to develop defect-free software practically. Software must go through exhaustive testing and debugging, but it requires time and money to enhance the reliability (Kumar and Ranjan 2017). Also due to increasing demand of software in today’s growing digital market, the occurrence of fault is inevitable in a software. Hence there should have some means to avoid software failures so that devastating losses whether related to life or any other field could be evaded.

According to IEEE Standard 729 (1983), reliability is the most significant quality aspect of the software. If we could measure the reliability of software under development, better we can predict whether the software would be operational in future or not. It will also help software developers, testers and project managers in estimating the right time to release the software and to predict the amount of required man-hour consumption during development of any software (Garg et al. 2011).

Software reliability models are one of the ways to simulate software reliability estimation curve to predict reliability of a system under study. Numerous reliability estimation models for software have been developed and all these are working on specific applications, specific environments, datasets and assumptions made by them. Then, what is the need for developing new software reliability models is a question in-front of researchers?

First need lies in the fact that, among the available research in software reliability model development, all developed models are basically made for reliability estimation of software’s developed under traditional waterfall SDLC process. Second, research publications are screening no more than 31% experimental researches and among this only 13% are purely experimental (Xavier et al. 2014). This low number is due to the reason that public experimental data sets in software reliability estimation are very narrow and producing reliability data through experimentation usually require long time cluster. Third, most generally developed models are NHPP based and very few are failure rate based models (Ortiz 1995; Moranda 1981; Littlewood and Sofer 1987; Luo et al. 2011; Mahapatra and Roy 2012; Zhang et al. 2003; Musa and Okumoto 1984; Huang et al. 2003; Pavlov et al. 2018; Roy et al. 2014; Singh et al. 2018; Wei et al. 2018). Failure rate-based models are among the earliest and most successful software reliability estimation models used in industries and academia. These models are grounded on JM software reliability model (Jelinski and Moranda 1972). They need further enhancement, so that more realistic assumptions like imperfect debugging and factors for exact reliability growth estimation can be incorporated into the model. Further, all existing software reliability models are developed under most basic waterfall SDLC process. Waterfall SDLC process for the development of software assumes that requirements from the end users are stable and it delivers whole software at one shot in the end (Boehm 1984; Model 2015). This may generate risks for the users as they do not have information till end what they will get. These limitations construct a need for the use of another methodology while modeling software reliability. Yet now, no research in software reliability assessment has been found that is based on latest SDLC process.

Software reliability is primarily dependent on the process of software development. The software developed using latest SDLC approach can capture and
implement all the user requirements within time and budget. Earlier waterfall SDLC process based reliability estimation models are very much stable and found to be unsuitable for the software industries because there are high risk and uncertainty of change in requirements. An iterative SDLC process (Basil and Turner 1975) is the latest methodology of software development that provides a way to the current methods of software development. It is a practical method of step-wise top-down refinement approach to the software development that replaces the waterfall SDLC process. It gathers user requirements in each iteration of software development and implements software with less uncertainty in risk and user satisfaction. Nowadays, methods of software development process are based on iterative software development process and these are like Rational Unified Process (RUP) (Kruchten 2004), Adaptive method (Larman 2004; Olage et al. 2007) (Agile methodology), XP (Extreme Programming) (Abrahamsson 2003), spiral model (Boehm 1986) and AZ (Akbar et al. 2018) development processes. These SDLC processes are considered as an aid of producing reliable software based on iterative model in development. Moreover, these are found to be used practically in industries and academia for software development and all these methods promotes iterative development process.

Further accurate software reliability estimation is dependent on the selection of optimum parameter values of models. The techniques for optimizing parameter values of the software reliability models are available through various classical methods of parameter estimation (Pham 2006; Myung 2003; Rana et al. 2013). These methods are based on number of constraints and may fall in local maxima and also do not converge to global maxima in multimodal cases. An alternative to these classical mathematical optimization methods are nature-inspired optimization algorithms for the solution of non-differential, non-linear and multi-modal problems (Tripathi et al. 2019; Wahab et al. 2015; Yang 2012, 2014).

Proposed failure rate model incorporates iterative SDLC process by replacing earlier waterfall based SDLC process. Further, it assumes imperfect debugging during each of iteration as there is always a possibility of fault introduction with feature addition in each of the iterations of software development. All latest iterative SDLC processes can be used to predict software reliability by applying proposed failure rate model. Existing failure rate model cannot be applied to the current software development methodology. Proposed model takes care of complexity and paradigm shift of iterative based software development process by introducing modulation factor.

Rest of the paper is schematized as follows: Sect. 2 exposes existing literature of the software reliability models and parameter estimation algorithms used for software reliability modeling. In Sect. 3 a new classification scheme for software reliability models based on iterative SDLC process is discussed. In Sect. 4 a new software reliability model is proposed. Sect. 5 discusses experimental setup and results of proposed model and the last section concludes the paper.
2 Literature review

For reliability estimation of the software more than 300 SRGMs have been developed under various categories of reliability estimation models like Pham (2006), Xie (1991), Ohba (1984), Yamada et al. (1983, 1986, 1992), Xiang et al. (2015), Kapur and Younes (1996), Kapur et al. (2012) and Sangeeta et al. (2020b). Experimental research work for software reliability estimation is found to be very few, reason being limited failure data and requisite of long time period. From the developed models, no model is well applicable in all type of applications and fits only with a specific application. Some of the model groups are having assumptions that are considered better and have higher predictive quality than other models. Most of the model development comprises NHPP based models and very few are centered on the failure rate behavior. Literature in this paper discusses well-known failure rate behavior-based models.

Jelinski–Moranda model (Jelinski and Moranda 1972) is the first software reliability estimation model. This model is perfect debugging based and assumes that there is fixed, constant and unknown number of initial faults in the software. The time between failures is assumed as independent and exponentially distributed. Shick–Wolverton (SW) model assumed that failure rate function is proportional to the current total number of faults and the time since last failure (Schick and Wolverton 1978). This idea has been evolved after modification from the basic JM model. Turk and Alsolami developed a model using JM model by applying Weibull distribution function for the amount of debugging time between fault occurrences (Turk and Alsolami 2016). Modified Shick–Wolverton model modified SW model by incorporating a cumulative amount of faults at a time (Pham 2006). These models incorporate perfect debugging process while making assumptions for the model development. Goel and Okumotto first time proposed imperfect debugging based model that incorporates imperfect debugging phenomenon by introducing fault removal probability (Pham 2006). JM Geometric Model modified JM model by assuming a geometric decrease in program failure rate at failure times (Moranda 1981). Littlewood and Sofer proposed a model with Bayesian modification in JM model and provided improvement for estimating the correct reliability of the software (Littlewood and Sofer 1987). Further modification to JM model based on the cloud theory has been applied by Luo et al. (2011). Chang and Liu proposed a generalized model by extending the JM model using a truncated distribution that matches up a self-exciting point process (Chang and Liu 2009). Mahapatra and Roy proposed a model and assumed the imperfect debugging behavior of the software with fault removal probability, probability of fault not removed and probability of fault introduction (Mahapatra and Roy 2012). This model introduced fault introduction probability while removing any fault during debugging. Sangeeta et al. proposed a new model for reliability estimation of open source software systems using a new testing effort function (Sangeeta et al. 2019). A new markov model has been proposed by Bharathi and Selvarani for software reliability estimation that has incorporated hidden markov approach for software reliability estimation with logic error (Bharathi and Selvarani 2020).
In failure rate behavior-based models, JM model assumed perfect debugging with single failure between time intervals after that multiple failure were proposed by SW but considers perfect debugging. Imperfect debugging based GOI model considers single failure in a time interval. Littlewood and Luo modified JM model according to the bayesian and cloud theory respectively. Mahapatra et al. introduced probability for fault introduction in the software. In these enhancements to the earliest failure rate models, multiple faults with imperfect debugging have not been covered by any model. These entire model enhancements are made for software’s developed under most basic waterfall development life cycle process. There is a need to incorporate latest software development processes for precise estimation of software reliability.

Parameter estimation process plays a crucial role in estimating accurate reliability of software product. There are number of models which are based on the classical methods for parameter estimation (Myung 2003). These are numerical technique based methods and these methods are doing well in a number of mathematical optimization problems. Nowadays meta-heuristic based optimization methods are found to be performing well in different domains. The algorithms like particle swarm optimization. Kennedy and Eberhart (1995) inspired by the bird flocks and genetic algorithm (Holland 1992) inspired by the natural evolution phenomenon have been well applied in various fields including the field of software reliability estimation. Sheta (2006) applied PSO for SRGM parameter estimation. Sangeeta et al. has proposed a magnetic navigation based optimizer for assessment of precise software reliability model parameters (Sharma et al. 2018). Jin and Jin (2016) applied improved swarm intelligent approach for parameter optimization of SRGM based on s-shaped testing effort function.

Hybrid nature-inspired algorithms are found to be performing well in various domains. Sangeeta et al. has developed an ecological space based hybrid swarm evolutionary algorithm for software reliability models parameter estimation (Sangeeta et al. 2020a). This paper hybridized the best features of two algorithms based on swarm and evolutionary phenomenon. Mirjalili and Hashim proposed a hybrid PSO-GSA algorithm by hybridizing PSO and gravitational search algorithm for mathematical function optimization (Mirjalili and Hashim 2010). Abraham et al. proposed a hybrid differential evolution and ABC algorithm (Abraham et al. 2012). Mirjalili and Wang proposed a binary optimization for hybrid PSO-GSA algorithm (Mirjalili et al. 2014). Sangeeta and Sitender has done a comprehensive analysis of hybrid nature-inspired algorithms for software reliability analysis and has evaluated several algorithms for software reliability estimation (Sangeeta and Sitender 2020). As hybrid algorithms are performing well in different fields, these algorithms could be applied in software reliability model parameter estimation so that accurate software reliability could be estimated.

3 Classification of software reliability models

This paper classifies software reliability models based on iterative SDLC process. Iterative model is like a cyclic process that mainly focuses on initial very simple implementation that progressively gains complexity and wider set of features till
the final system is completed (Kapur and Garg 1991; Huang and Lyu 2005; Greer and Ruhe 2004). Most recently built iteration and its feedback from evaluation are used in the next iteration, and accordingly, refinements are made in future iterations (Wong et al. 2016; Larman and Basili 2003). Each of iteration provides enhancements and at-least found to be better than the last. Iterative model adapts rapidly to ever-changing needs of projects and clients within lowest time and budget. Sharma et al. (2010) has given classification of software reliability models according to waterfall SDLC process (McCall et al. 1987; Smidts et al. 1998; Goševa-Popstojanova and Trivedi 2001; Wang et al. 1999; Gokhale et al. 1996, 1998, 2004; Gokhale and Trivedi 1999; Weiss and Weyuker 1988; Littlewood 1979). In this paper, keeping iterative software development practice in mind, software reliability models are categorized in Fig. 1. Every phase in iterative software development process is associated with specific requirements and future plans. Figure 1 shows that each of the iteration is associated with a modulation parameter showing the added functionality and user acceptance level in each iteration. Initially, system is least reliable but as the number of iterations proceeds, system moves towards refinement.

Fig. 1 Classification of software reliability models based on iterative approach
and gains reliability. Models are grouped into five categories. Each model category is assigned to a specific phase in iterative software development process for reliability estimation.

4 Proposed model

Keeping new SDLC processes and technologies, a new failure rate model centered on iterative SDLC process has been proposed. The proposed model has been derived from the general class of failure rate behavior-based models and exploits iterative behavior of software development process. In iterative process of software development, entire software is built and delivered to the end user in iterations. The process starts with simple implementation of key samples of a problem and iteratively enhances existing software releases until full system is implemented (Larman and Basili 2003). At each of iteration release, feedback from the iteration is available for next iteration (Kapur and Garg 1991; Huang and Lyu 2005; Greer and Ruhe 2004). Feedback is mainly about the functionality and user aspect of the software. At each released stages of software, not only extensions but design modifications may be made. Each of the iteration makes step wise refinements in an effective way to converge to the full implementation of a problem. Main focus of feedback analysis is to find the amount of defects in an iteration that gets injected in the upcoming iteration (Wong et al. 2016). Using feedback in each iteration, analysis of changing needs in an up-coming iteration can be made. These changing needs in each iteration are essential to know because these tells information about how much more defects may occur and how much more effort and functionality is needed in making extensions and design modifications to the released iteration.

In modeling software reliability, there is a need to introduce a factor that will engulf all the changing needs depending on the defect analysis in each of iteration. The varying needs in each of iteration of software development include bug reports, additional functionalities and testing effort required to find the amount of defects that gets injected/removed in each upcoming iteration. These changing needs are incorporated in proposed failure rate model using modulation factor $\gamma$ given in (1), so that precise reliability growth of the system could be estimated. The proposed model assumes that fault removal process is imperfect. Due to imperfect debugging, regenerated faults are induced in successive iterations.

4.1 Proposed model assumptions

- Initial software fault is unknown and constant in an iteration.
- Each fault in an iteration is independent and it may be equally likely to cause a failure while testing.
- The interval of time between fault occurrences in each iteration is independent and follows an exponential distribution.
- Software failure rate remains constant over the intervals between fault occurrences.
The software failure rate is proportional to the number of faults that remain in the software and modulation factor $\gamma$.

In each iteration the detected fault is removed with a probability $p$, not removed perfectly with a probability $q$ and new fault may be introduced with a probability $r$. Here $p + q + r = 1$ and the probability $p > r$.

Faults are injected from previous iterations along with newly introduced faults in the current iteration. Newly induced faults are caused by added and modified functionalities in a respective iteration of software development. Depending on the number of initial iterative faults, there is a need to modify the amount of resources allocated for debugging in each iteration. The modulation factor $\gamma$ as defined in (1) reflects the modified needs that integrate iterative development processes in software failure rate models. Modulation factor changes its value according to the modulation parameter $\mu$ as shown in (1). Moreover, changing needs in each iteration is different and vary according to (1).

$$\gamma = \mu + (1 - \mu)/\mu, \quad (0 < \mu \leq 1.0).$$

Here $\mu$ is the modulation parameter that represents newly added functionality and user acceptance in the current iteration. Its value is almost 0 at the beginning and becomes 1.0 in the final iteration of iterative software development process. Modulation parameter $\mu$ takes its value from 0 to 1.0 by assuming that with growing number of iterations, level of user acceptance increases from lower to higher. It shows abrupt changes in its value in initial iterations due to preliminary design changes, testing effort, and user acceptance. When it’s values reaches near to one then software under development is assumed to be reliable enough and has achieved all the required functionalities to fulfill the end user needs.

### 4.2 Model formulation

Failure rate function ($\lambda(t_i)$) with imperfect debugging is modeled in (2).

$$\lambda(t_i) = \phi \left( N - \frac{(n_{i-1})(\gamma)}{(i - 1)} (p - r) \right), \quad i = 1, 2, \ldots, N$$

where $\gamma$: modulation factor for representing changing needs in each of the iteration of software development; $n_{i-1}$: cumulative number of failures at $(i - 1)$th failure interval; $N$: number of initial faults in software; $\phi$: proportionality constant.

Cumulative density function $F(t_i)$ and reliability function $R(t_i)$ is calculated in (3) and (4).

$$F(t_i) = 1 - e^{-\phi[N - \frac{(n_{i-1})(\gamma)}{(i - 1)} (p - r)]\mu]}$$

$$R(t_i) = e^{-\phi[N - \frac{(n_{i-1})(\gamma)}{(i - 1)} (p - r)]\mu]}.$$
Following a variation of $\gamma$ in (5) and considering $p$ being the probability of fault removal and $r$ as the fault introduction probability, proposed model behaves as the GS Mahapatra et al. model.

4.3 Parameter estimation

In the proposed model there are three unknown parameters $N$, $n$ and $\phi$, these parameters are estimated at different values of $\gamma$. MLE has been used to estimate the values of the parameters. Parameter estimation by MLE method requires solutions of complex equations by maximizing the likelihood of model parameters. Probability density function $f(t_i)$ for the proposed model is given in (6).

$$f(t_i) = \phi \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] \cdot e^{-\phi \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] t_i}$$  \hspace{1cm} (6)

The likelihood function $L(N)$ is calculated in (6) using (7).

$$L(N) = \prod_{i=1}^{n} f(t_i) = \prod_{i=1}^{n} \left[ \phi \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] e^{-\phi \sum_{i=1}^{n} \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] t_i} \right].$$  \hspace{1cm} (7)

Taking the log of $L(N)$, LLF is calculated in (8).

$$LLF = \ln L(N) = n \ln \phi + \sum_{i=1}^{n} \ln \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right]$$

\hspace{1cm} $$- \sum_{i=1}^{n} \phi \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] t_i.$$  \hspace{1cm} (8)

Solution using log likelihood function for parameter estimation involves calculation of its partial derivatives with respect to $N$, $n$ and $\phi$ respectively and then equating them to value zero. MLE of the parameters are calculated from (9), (10) and (11).

$$\frac{n}{\phi} = \sum_{i=1}^{n} \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] t_i$$  \hspace{1cm} (9)

$$n = \phi \sum_{i=1}^{n} \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right] t_i$$  \hspace{1cm} (10)

$$\frac{1}{\sum_{i=1}^{n} \left[ N - \frac{(n_{i-1})\gamma}{i-1} (p - r) \right]} = \phi \sum_{i=1}^{n} t_i.$$  \hspace{1cm} (11)
5 Experimental setup and results

Suitability of the proposed model has been tested using Tera Promise repository bug report files of Eclipse (DS1) and JDT (DS2) open source software at https://zenodo.org/record/268486#.W-QsPpMzY2w. These data sets have been given by An Ngoc Lam. The bug reports contain table having contents as bug, d, summary, description, report time, report time status, commit, and commit time files. Datasets have been extracted from these bug reports and reformatted in time domain format. DS1 includes eight minor releases and four major releases starting from the year 2001 to the year 2013. DS2 includes 3 major releases and 6 minor releases starting from the year 2002 to the year 2014. Model parameters include $N, n, \phi, \gamma, p$ and $r$. Fault removal probability $p$ and fault introduction probability $r$ cannot be estimated from DS1 and DS2 and depends on the project type and skill set of persons involved in development and testing. Based on these two factors fault removed during testing is assumed 95% and fault introduced is assumed 3%. Parameters $N, n, \phi, \gamma$ are estimated using hybrid PSO-GSA algorithm and MLE technique in (6), (7) and (8) to maximize the log-likelihood function value using (9), (10) and (11). The goodness of fit for the proposed model is measured using SSE and MSE for each application datasets (Sharma et al. 2010). These statistics are used for comparison of the proposed model with existing failure rate models.

The values of $\mu$ with respect to iteration are different for different software projects and depend on the type of project and user acceptance levels. Figure 2 depicts variation in $\mu$ values with respect to number of iterations for DS1. The value of $\mu$ decreases in successive iterations 1.0 to 2.0, 2.1 to 3.0 and 3.3 to 3.4 by difference of 0.064, 0.081 and 0.045 respectively. These little deeps in $\mu$ values represents variation in compliance of functionality requirements and lower values of user acceptance levels for eclipse software dataset (DS1). Overall values of $\mu$ shows increasing trends with successive iteration as shown in Fig. 2 and finally it reaches near to 1.0 in final iteration release.

Figure 3 is revealing the change in functionalities and its corresponding increase in user acceptance for DS2 of JDT software product. At each of the iteration release there is increase in user acceptance level and it is illustrated with the values of $\mu$ in all successive iterations. For major iteration releases the value of $\mu$ are 0.0869, 0.1108 and 0.2144, for minor iteration these values are 0.1381, 0.2684, 0.3677, 0.6787, 0.7241 and 0.9539. The user acceptance increases in all iterations but a large variation of 0.31102 in user acceptance level is found in 3.4 to 3.5 iteration. It represents enhanced functionality and accomplishment of all requirements at end-users. Overall values of $\mu$ in successive iterations are reflecting variation in functionalities and user acceptance level in Fig. 3 that increases by small amount in each iteration and finally reaches near to 1.0 in last iteration release.
Table 1  Summary of failure rate based software reliability models

| Sr. No. | Model name                                                         | Failure intensity                                      |
|---------|-------------------------------------------------------------------|--------------------------------------------------------|
| 1       | Jelinski–Moranda model (JM) (Jelinski and Moranda 1972)          | $\lambda(t_i) = \phi[N - (i - 1)]$                    |
| 2       | Schick & Wolverton model (SW) (Schick and Wolverton 1978)        | $\lambda(t_i) = \phi[N - (i - 1)] t_i$                |
| 3       | Goel Okumotto imperfect debugging model (GOI) (Pham 2006)        | $\lambda(t_i) = \phi[N - p(i - 1)]$                   |
| 4       | Mahapatra et al. model (Mahapatra and Roy 2012)                  | $\lambda(t_i) = \phi[N - p(i - 1) + r(i - 1)]$        |
| 5       | Modified S–W model (MSW) (Pham 2006)                             | $\lambda(t_i) = \phi[N - (n_{i-1})I_t]$               |
| 6       | Proposed model                                                   | $\lambda(t_i) = \phi[N - \frac{n_{i-1}+r}{(i-1)}(p - r)]$|

Fig. 2  Plot of $\mu$ versus iteration for DS1

Fig. 3  Plot of $\mu$ versus iteration for DS2
5.1 Result analysis

5.1.1 CASE 1: eclipse software dataset (DS1)

In this section, the goodness-of-fit of the proposed model is calculated and compared with existing models given in Table 1 for DS1. The value of $\gamma$ for major iterative release 1.0, 2.0, 3.0 and 4.1 are 2.2666, 2.8763, 3.7891 and 1.1258 respectively. In minor iteration release, the value of $\gamma$ for iterative release 2.1, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6 and 4.2 are 2.6343, 3.2217, 2.5788, 2.3121, 2.7295, 2.6434, 2.0435 and 1.00009 respectively. The value of $\gamma$ shows large changes in successive major iterations as compared to successive minor iterations due to varying needs in major and minor iterations. The goodness of fit of the models is shown in Table 2. From the analysis of results, it is found that proposed model fits well in term of SSE in all iterations of DS1 except at iteration number 3.1 where GOI model, GS Mahapatra model and SW model outperforms proposed model. The proposed model has clear-cut outperform all models under comparison in 11 iterations for DS1. It shows the lowest values of SSE in 91.6% iterations. In term of MSE proposed model is winner in nine iterations. In iteration 2.1 MSW model is performing well than the proposed model. In iteration 3.3, GOI model, SW model, and GS Mahapatra model are performing well than the proposed model. There is a tie among proposed model and GS Mahapatra model in iteration 2.0, where both of them outperform all other models. Proposed model efficiency in terms of iterations is plotted in Fig. 4. The proposed model has clear-cut outperform other models in 75% iterations by achieving lowest value of MSE. Result shows that proposed model has given a significantly better fit to iterative data by adapting according to varying needs of different iterations. The evaluation and comparison of goodness-of-fit of all models given in Table 1 in terms of SSE and MSE for all iterations shows that the proposed model has promising technical merit in the sense that it provides development terms with both iterative SDLC requirements and traditional reliability measures.

![Efficiency of proposed model in terms of iterations for DS1](image)
5.1.2 CASE 2: JDT dataset

To test the applicability of the proposed model in this case dataset DS2 is used. Table 3 is showing the estimated values of model parameters and goodness of fit criteria. The value of $\gamma$ for major iterative release 1.4, 2.0 and 3.2 are 10.5954, 8.1358 and 3.8778 respectively. In minor iteration releases the value of $\gamma$ for iterative release 2.1, 3.3, 3.4, 3.5, 3.6 and 3.7 are 6.3815, 2.9946, 2.0872, 1.1521, 1.1052 and 1.0022, respectively. These values of $\gamma$ are showing large deviations in successive major releases as compared to successive minor releases due to the varying needs in each of the major and minor iterations. Table 3 is showing the goodness-of-fit measures of all models depicted in Table 1 for DS2. Out of total nine iterations in DS2, proposed model outperform other models under comparison in eight iterations, in terms of SSE values. JM model performs well than proposed model in iteration 3.4. Proposed model attains lower-most value of SSE in 88.8% of iterations. In terms of MSE values for nine iterations, proposed model out-performs other models in six iterations. In iteration 2.0, GS Mahapatra model and GOI model are performing better than the proposed model. In iteration 3.7, proposed model outperforms JM, GOI, SW and MSW models except GS Mahapatra model. The proposed model outperforms GOI, SW, GS Mahapatra and MSW model except JM model in iteration number 3.6. The proposed model has performed better than all other models under comparison in 66.6% iterations by attaining the minimum value of MSE. Efficiency of proposed model in terms of iteration is plotted in Fig. 5. Result analysis in Table 3 shows that the proposed model has significantly better fit to iterative data by fine-tuning to varying needs of different iterations.

![Fig. 5 Proposed model efficiency in terms of iteration for DS2](image-url)
Table 2: Goodness-of-fit estimated using DS1 (eclipse software failure dataset)

| Sr. No. | Model                  | Iteration 1.0 | Iteration 2.0 | Iteration 2.1 |
|---------|------------------------|---------------|---------------|---------------|
|         |                        | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE |
| 1       | JM model               | $\phi = 5.74E-06$, $N = 4$ | 5.23 | 5   | $\phi = 2.68E-05$, $N = 8$ | 375.51 | 17.05 | $\phi = 5.039E-06$, $N = 27$ | 452.2 | 16.74 |
| 2       | GOI model              | $\phi = 1.64E-05$, $N = 3$ | 1.51 | 1   | $\phi = 2.89E-06$, $N = 28$ | 92.01 | 4.18 | $\phi = 1.95E-06$, $N = 26$ | 133.34 | 4.93 |
| 3       | SW model               | $\phi = 2.124E-06$, $N = 3$ | 17.31 | 17.00 | $\phi = 4.70E-06$, $N = 28$ | 218.81 | 9.91 | $\phi = 3.83E-06$, $N = 29$ | 126.6 | 4.66 |
| 4       | GS Mahapatra model     | $\phi = 7.77E-07$, $N = 3$ | 1.26 | 1   | $\phi = 2.53E-05$, $N = 23$ | 55.46 | 2.5   | $\phi = 3.20E-06$, $N = 28$ | 131.31 | 4.85 |
| 5       | MSW model              | $\phi = 2.23E-06$, $N = 2$, $n = 58$ | 5.0 | 4.9 | $\phi = 1.52E-06$, $N = 9$, $n = 209$ | 528.7 | 15.89 | $\phi = 1.36E-05$, $N = 18$, $n = 316$ | 118.67 | 3.89 |
| 6       | Proposed model         | $\phi = 2.86E-05$, $N = 5$, $n = 11$, $\gamma = 2.2666$ | 1.24 | 0.66 | $\phi = 4.70E-06$, $N = 27$, $n = 362$, $\gamma = 2.8763$ | 45.12 | 2.5   | $\phi = 1.67E-06$, $N = 30$, $n = 415$, $\gamma = 2.6343$ | 101.02 | 4.39 |

| Sr. No. | Model                  | Iteration 3.0 | Iteration 3.1 | Iteration 3.2 |
|---------|------------------------|---------------|---------------|---------------|
|         |                        | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE |
| 1       | JM model               | $\phi = 2.40E-06$, $N = 124$ | 56,948 | 605.82 | $\phi = 2.99E-05$, $N = 100$ | 609.1 | 45.45 | $\phi = 2.21E-06$, $N = 122$ | 13,940.73 | 119.14 |
| 2       | GOI model              | $\phi = 1.37E-06$, $N = 100$ | 45,510.1 | 559.22 | $\phi = 3.07E-07$, $N = 126$ | 1018.8 | 7.59 | $\phi = 8.61E-06$, $N = 106$ | 7246.05 | 61.93 |
| 3       | SW model               | $\phi = 2.92E-05$, $N = 121$ | 39,981 | 494.69 | $\phi = 2.99E-05$, $N = 125$ | 1123 | 8.38 | $\phi = 5.40E-06$, $N = 90$ | 12,585 | 107.56 |
| 4       | GS Mahapatra model     | $\phi = 1.25E-06$, $N = 125$ | 55,122.2 | 586.40 | $\phi = 6.62E-06$, $N = 130$ | 1015 | 7.57 | $\phi = 1.25E-06$, $N = 115$ | 7794.28 | 68.97 |
| 5       | MSW model              | $\phi = 1.29E-06$, $N = 123$, $n = 298$ | 66,890.4 | 679.32 | $\phi = 1.36E-05$, $N = 135$, $n = 4979$ | 2000.7 | 15.9 | $\phi = 2.08E-06$, $N = 132$, $n = 4108$ | 9685.12 | 89.67 |
| 6       | Proposed model         | $\phi = 2.98E-05$, $N = 102$, $n = 5690$, $\gamma = 3.7891$ | 39,705.2 | 484.14 | $\phi = 2.96E-05$, $N = 136$, $n = 6132$, $\gamma = 3.2217$ | 1589.82 | 12.22 | $\phi = 2.98E-05$, $N = 122$, $n = 6910$, $\gamma = 2.5788$ | 952.2 | 8.14 |

| Sr. No. | Model                  | Iteration 3.3 | Iteration 3.4 | Iteration 3.5 |
|---------|------------------------|---------------|---------------|---------------|
|         |                        | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE |
| 1       | JM model               | $\phi = 2.95E-05$, $N = 117$ | 6810.14 | 58.24 | $\phi = 8.28E-06$, $N = 53$ | 1153.32 | 23.53 | $\phi = 1.63E-06$, $N = 30$ | 373.3 | 15.54 |
| 2       | GOI model              | $\phi = 1.25E-06$, $N = 115$ | 9371.1 | 80.09 | $\phi = 3.06E-06$, $N = 50$ | 1545.41 | 1320.9 | $\phi = 3.81E-06$, $N = 29$ | 137.74 | 5.71 |
| 3       | SW model               | $\phi = 2.95E-05$, $N = 123$ | 7318.81 | 62.55 | $\phi = 2.93E-05$, $N = 54$ | 1153.73 | 23.53 | $\phi = 1.70E-07$, $N = 29$ | 551.1 | 22.96 |
| 4       | GS Mahapatra model     | $\phi = 3.06E-06$, $N = 114$ | 2217.4 | 18.99 | $\phi = 1.63E-05$, $N = 55$ | 905.4 | 18.46 | $\phi = 3.81E-06$, $N = 29$ | 137.8 | 5.71 |
Table 2 (continued)

| Sr. No. | Model         | Iteration 3.3 | Iteration 3.4 | Iteration 3.5 |
|---------|---------------|---------------|---------------|---------------|
|         |               | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE |
| 5       | MSW model     | $\phi = 5.31E^{-06}$, $N = 121$, $n = 3129$ | 2903.9 | 27.87 | $\phi = 1.72E^{-05}$, $N = 52$, $n = 2094$ | 3589.96 | 29.79 | $\phi = 1.26E^{-05}$, $N = 17$, $n = 2663$ | 489.74 | 18.24 |
| 6       | Proposed model| $\phi = 2.96E^{-05}$, $N = 123$, $n = 7879$, $\gamma = 2.3121$ | 2203.05 | 19.49 | $\phi = 2.92E^{-05}$, $N = 53$, $n = 2094$, $\gamma = 2.7295$ | 520.23 | 11.55 | $\phi = 2.82E^{-05}$, $N = 25$, $n = 416$, $\gamma = 2.6434$ | 86.62 | 4.3  |

| Sr. No. | Model         | Iteration 3.6 | Iteration 4.1 | Iteration 4.2 |
|---------|---------------|---------------|---------------|---------------|
|         |               | Estimated Parameter values | SSE | MSE | Estimated Parameter values | SSE | MSE | Estimated Parameter values | SSE | MSE |
| 1       | JM model      | $\phi = 6.24E^{-07}$, $N = 28$ | 199.91 | 7.65 | $\phi = 2.20E^{-05}$, $N = 19$ | 286.65 | 22 | $\phi = 1.92E^{-05}$, $N = 31$ | 441.12 | 15.21 |
| 2       | GOI model     | $\phi = 3.20E^{-06}$, $N = 28$ | 160.01 | 6.15 | $\phi = 1.58E^{-05}$, $N = 17$ | 111.21 | 8.54 | $\phi = 3.20E^{-06}$, $N = 30$ | 121.21 | 4.17 |
| 3       | SW Model      | $\phi = 3.74E^{-06}$, $N = 27$ | 154.54 | 5.92 | $\phi = 1.93E^{-05}$, $N = 19$ | 284.86 | 22.1 | $\phi = 6.50E^{-06}$, $N = 31$ | 193.34 | 6.66 |
| 4       | GS Mahapatra model | $\phi = 6.65E^{-07}$, $N = 28$ | 152.23 | 5.86 | $\phi = 5.53E^{-06}$, $N = 17$ | 129 | 9.92 | $\phi = 7.29E^{-07}$, $N = 31$ | 166.65 | 5.72 |
| 5       | MSW model     | $\phi = 2.97E^{-05}$, $N = 17$, $n = 2624$ | 315.00 | 14.92 | $\phi = 2.68E^{-05}$, $N = 8$, $n = 1529$ | 178.22 | 7.34 | $\phi = 2.97E^{-05}$, $N = 30$, $n = 3729$ | 704.86 | 26.87 |
| 6       | Proposed model| $\phi = 2.94E^{-05}$, $N = 30$, $n = 401$, $\gamma = 2.0435$ | 91.15 | 4.14 | $\phi = 2.87E^{-05}$, $N = 16$, $n = 136$, $\gamma = 1.1258$ | 49.95 | 5.44 | $\phi = 2.82E^{-05}$, $N = 32$, $n = 513$, $\gamma = 1.00009$ | 72.24 | 2.88 |
| Sr. No. | Model                      | Iteration 1.4 | Iteration 2.0 | Iteration 2.1 |
|--------|----------------------------|---------------|---------------|---------------|
|        | Estimated parameter values | SSE   | MSE   | Estimated parameter values | SSE   | MSE   | Estimated parameter values | SSE   | MSE   |
| 1      | JM model                   | $\phi = 1.46E-05$, N = 22 | 1510 | 62.91 | $\phi = 5.63E-08$, N = 13 | 1792.56 | 66.37 | $\phi = 2.81E-05$, N = 37 | 2638.12 | 203.02 |
| 2      | GOI model                  | $\phi = 4.11E-06$, N = 15 | 1517 | 75.85 | $\phi = 4.80E-08$, N = 18 | 1486.81 | 55.04 | $\phi = 5.33E-06$, N = 35 | 1650.04 | 127.9 |
| 3      | SW model                   | $\phi = 8.18E-07$, N = 9 | 2165 | 90.21 | $\phi = 6.99E-06$, N = 17 | 2427.45 | 187.9 | $\phi = 5.36E-06$, N = 38 | 3278.45 | 252.87 |
| 4      | GS Mahapatra model         | $\phi = 9.00E-06$, N = 12 | 1871 | 77.96 | $\phi = 1.08E-06$, N = 16 | 1434.79 | 53.11 | $\phi = 1.21E-06$, N = 57 | 3808.85 | 293.48 |
| 5      | MSW model                  | $\phi = 7.43E-06$, N = 19, n = 242.98 | 2909.88 | 121.09 | $\phi = 1.62E-05$, N = 15, n = 282.93 | 1870.67 | 69.26 | $\phi = 8.73E-06$, N = 29, n = 2569.9 | 1741.90 | 164.48 |
| 6      | Proposed model             | $\phi = 2.99E-05$, N = 13, n = 160, $\gamma = 10.5954$ | 1379 | 57.54 | $\phi = 2.97E-05$, N = 17, n = 199, $\gamma = 8.1358$ | 1292 | 56.34 | $\phi = 2.82E-05$, N = 37, n = 234, $\gamma = 6.3815$ | 683.70 | 75.88 |

| Sr. No. | Model                      | Iteration 3.2 | Iteration 3.3 | Iteration 3.4 |
|--------|----------------------------|---------------|---------------|---------------|
|        | Estimated parameter values | SSE   | MSE   | Estimated parameter values | SSE   | MSE   | Estimated parameter values | SSE   | MSE   |
| 1      | JM Model                   | $\phi = 2.35E-06$, N = 50 | 179.79 | 11.19 | $\phi = 2.78E-05$, N = 9 | 404 | 33.666 | $\phi = 8.86E-06$, N = 18 | 37 | 12.33 |
| 2      | GOI model                  | $\phi = 8.16E-07$, N = 56 | 317.67 | 5.56 | $\phi = 1.48E-05$, N = 16 | 989.9 | 61.825 | $\phi = 2.35E-06$, N = 18 | 206.56 | 17.16 |
| 3      | SW model                   | $\phi = 2.17E-06$, N = 58 | 1056 | 15.68 | $\phi = 4.70E-06$, N = 8 | 589.1 | 40.78 | $\phi = 3.83E-06$, N = 16 | 750.9 | 68.00 |
| 4      | GS Mahapatra model         | $\phi = 1.21E-06$, N = 57 | 56,500.09 | 991.23 | $\phi = 8.92E-06$, N = 26 | 604.01 | 37.75 | $\phi = 1.03E-05$, N = 28 | 1241.01 | 103.41 |
| 5      | MSW model                  | $\phi = 2.91E-05$, N = 52, n = 1723 | 2989.06 | 29.69 | $\phi = 2.99E-05$, N = 9, n = 1028 | 486.32 | 33.960 | $\phi = 2.91E-05$, N = 13, n = 1124 | 1839.56 | 146.92 |
| 6      | Proposed model             | $\phi = 2.91E-05$, N = 60, n = 1788, $\gamma = 3.8778$ | 171.89 | 3.226 | $\phi = 2.97E-05$, N = 19, n = 179, $\gamma = 2.9946$ | 31 | 2.583 | $\phi = 2.92E-05$, N = 18, n = 119, $\gamma = 2.0871$ | 91 | 11.37 |
| Sr. No. | Model                  | Iteration 3.5 | Iteration 3.6 | Iteration 3.7 |
|---------|------------------------|--------------|--------------|--------------|
|         |                        | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE | Estimated parameter values | SSE | MSE |
| 1       | JM model               | $\phi = 2.49E^{-05}$, $N = 8$ | 57 | 4.384 | $\phi = 2.93E^{-05}$, $N = 4$ | 76 | 7.6 | $\phi = 1.50E^{-06}$, $N = 16$ | 280 | 45.02 |
| 2       | GOI model              | $\phi = 1.06E^{-06}$, $N = 13$ | 76.53 | 25.33 | $\phi = 2.01E^{-05}$, $N = 11$ | 46,721 | 359.39 | $\phi = 1.58E^{-06}$, $N = 13$ | 90.011 | 9.01 |
| 3       | SW model               | $\phi = 2.92E^{-05}$, $N = 10$ | 3019.9 | 119.05 | $\phi = 2.99E^{-05}$, $N = 29$ | 12,089 | 367.83 | $\phi = 5.40E^{-06}$, $N = 26$ | 200.9 | 11.08 |
| 4       | GS Mahapatra model     | $\phi = 4.78E^{-07}$, $N = 5$ | 38.225 | 318.54 | $\phi = 1.26E^{-05}$, $N = 85$ | 11,534 | 384.48 | $\phi = 5.30E^{-06}$, $N = 12$ | 952 | 8.14 |
| 5       | MSW model              | $\phi = 2.31E^{-05}$, $N = 3$, $n = 137$ | 20,687 | 156.94 | $\phi = 1.34E^{-05}$, $N = 15$, $n = 134$ | 34,834.09 | 329.21 | $\phi = 2.69E^{-05}$, $N = 14$, $n = 1599$ | 1109.23 | 57.34 |
| 6       | Proposed model         | $\phi = 2.96E^{-05}$, $N = 7$, $n = 25, \gamma = 1.1521$ | 18 | -24.002 | $\phi = 2.67E^{-05}$, $N = 18$, $n = 171, \gamma = 1.1051$ | 192 | 21.33 | $\phi = 2.74E^{-05}$, $N = 15$, $n = 78, \gamma = 1.0022$ | 54 | 9.00 |
6 Conclusion

Failure rate models available in literature are centered on most traditional waterfall SDLC process. However, new software development processes have been developed and found to be more beneficial than waterfall SDLC process like iterative life cycle processes. Keeping in view, new software development environments and technologies, a new failure rate model by exploiting iterative behavior of software development process is proposed. The changing needs in each of the iteration are reflected in the proposed model using a modulation factor. Calculated values of $\gamma$ parameter are significantly reflecting all changing requirements for each upcoming iterations numerically. These values are meaningfully representing how much impact is of adding and removing new features with the level of user acceptance in each of upcoming iteration. In order to compare the performance of the proposed model, five well-known software failure rate models JM, GOI, SW, GS Mahapatra and MSW have been applied for dataset DS1 and DS2. The proposed model is a clear-cut winner in 11 iterations in SSE and 9 iterations in MSE out of 12 iterations and 8 iteration in SSE and 6 iterations in MSE out of 9 iterations for DS1 and DS2 respectively. Overall, in 83.33% of iterations for DS1 and 77.77% of iterations for DS2, the proposed model has shown better results in terms of goodness of fit by successfully incorporating varying needs in each of iteration. The data collected from real applications and comparison of goodness of fit shows that the proposed model successfully incorporated varying needs in each of iteration and performed better than JM, GOI, SW, GS Mahapatra and MSW models.

Author's contribution  All authors in this manuscript contributes equally.

Funding  No funding information is available for this manuscript.

Declarations

Conflict of interest  We have no conflicts of interest to disclose.

Ethics approval  This article does not contain any studies with animals performed by any of the authors. This article does not contain any studies with human participants or animals performed by any of the authors.

Consent for publication  We give our consent for the publication of this article in your esteemed journal.

References

IEEE standard glossary of software engineering terminology. In: ANSI/IEEE Std 729-1983, pp.1–40. IEEE (1983), https://doi.org/10.1109/ieestd.1983.7435207
Abraham, A., Jatoth, R.K., Rajasekhar, A.: Hybrid differential artificial bee colony algorithm. J. Comput. Theor. Nanosci. 9(2), 249–257 (2012), https://doi.org/10.1166/jctn.2012.2019
Abrahamsson, P.: Extreme programming: first results from a controlled case study. In: Proceedings of the 20th IEEE Instrumentation Technology Conference (Cat No 03CH37412) EURMIC-03, IEEE. https://doi.org/10.1109/eurmic.2003.1231599 (2003)
Akbar, M.A., Sang, J., Khan, A.A., Fazal-E-Amin, Nasrullah, Shaﬁq, M., Hussain, S., Hu, H., Elahi, M., Xiang, H.: Improving the quality of software development process by introducing a new methodology—AZ-model. IEEE Access 6, 4811–4823 (2018). https://doi.org/10.1109/access.2017.2787981

Basil, V.R., Turner, A.J.: Iterative enhancement: a practical technique for software development. IEEE Trans. Softw. Eng. 10(4), 390–396 (1975). https://doi.org/10.1109/tse.1975.6312870

Bharathi, R., Selvarani, R.: Hidden Markov model approach for software reliability estimation with logic error. Int. J. Autom. Comput. 17, 1–16 (2020)

Boehm, B.W.: Software engineering economics. IEEE Trans. Softw. Eng. 1, 4–21 (1984)

Boehm, B.: A spiral model of software development and enhancement. ACM SIGSOFT Softw. Eng. Notes 11(4), 14–24 (1986). https://doi.org/10.1145/12944.12948

Chang, Y.C., Liu, C.T.: A generalized JM model with applications to imperfect debugging in software reliability. Appl. Math. Modell. 33(9), 3578–3588 (2009). https://doi.org/10.1016/j.apm.2008.11.018

Garg, R.K., Sharma, K., Nagpal, C.K., Garg, R., Garg, R., Kumar, R., Sandhya: Ranking of software engineering metrics by fuzzy-based matrix methodology. Softw. Test. Verif. Reliab. 23(2), 149–168 (2011). https://doi.org/10.1002/stvr.459

Gokhale, S.S., Trivedi, K.S.: A time/structure based software reliability model. Ann. Softw. Eng. 8(1/4), 85–121 (1999). https://doi.org/10.1023/a:1018923329647

Gokhale, S.S., Marinos, P.N., Trivedi, K.S.: Important milestones in software reliability modeling. Tech. rep., North Carolina State University. Center for Advanced Computing and Communication (1996)

Gokhale, S., Wong, W., Trivedi, K., Horgan, J.: An analytical approach to architecture-based software reliability prediction. In: Proceedings. IEEE International Computer Performance and Dependability Symposium. IPDS-98 (Cat. No.98TB100248). IEEE Computer Society (1998). https://doi.org/10.1109/ipds.1998.707705

Gokhale, S.S., Lyu, M.R., Trivedi, K.S.: Analysis of software fault removal policies using a non-homogeneous continuous time Markov chain. Softw. Qual. J. 12(3), 211–230 (2004). https://doi.org/10.1023/b: sqjo.0000034709.63615.8b

Goševa-Popstojanova, K., Trivedi, K.S.: Architecture-based approach to reliability assessment of software systems. Perform. Eval. 45(2–3), 179–204 (2001)

Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Inf. Softw. Technol. 46(4), 243–253 (2004). https://doi.org/10.1016/j.infsof.2003.07.002

Holland, J.H.: Genetic algorithms. Sci. Am. 267(1), 66–72 (1992). https://doi.org/10.1038/scientificamerican0792-66

Huang, C.Y., Lyu, M.: Optimal release time for software systems considering cost, testing-effort, and test efficiency. IEEE Trans. Reliab. 54(4), 583–591 (2005). https://doi.org/10.1109/tr.2005.859230

Huang, C.Y., Lyu, M., Kuo, S.Y.: A unified scheme of some nonhomogenous poisson process models for software reliability estimation. IEEE Trans. Softw. Eng. 29(3), 261–269 (2003). https://doi.org/10.1109/tse.2003.1183936

Jelinski, Z., Moranda, P.: Software reliability research. In: Statistical Computer Performance Evaluation, pp. 465–484. Elsevier, Amsterdam (1972). https://doi.org/10.1016/b978-0-12-266950-7.50028-1

Jin, C., Jin, S.W.: Parameter optimization of software reliability growth model with s-shaped testing-effort function using improved swarm intelligent optimization. Appl. Soft Comput. 40, 283–291 (2016). https://doi.org/10.1016/j.asoc.2015.11.041

Kapur, P.K., Garg, R.B.: Optimal release policies for software systems with testing effort. Int. J. Syst. Sci. 22(9), 1563–1571 (1991). https://doi.org/10.1080/00207729108910731

Kapur, P., Younes, S.: Modelling an imperfect debugging phenomenon in software reliability. Microelectron. Reliab. 36(5), 645–650 (1996)

Kapur, P.K., Pham, H., Aggarwal, A.G., Kaur, G.: Two dimensional multi-release software reliability modeling and optimal release planning. IEEE Trans. Reliab. 61(3), 758–768 (2012). https://doi.org/10.1109/tr.2012.2207531

Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of International Conference on Neural Networks ICNN’95. IEEE (1995). https://doi.org/10.1109/icnn.1995.488968

Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley Professional, Boston (2004)

Kumar, S., Ranjan, P.: A phase wise approach for fault identification. J. Inf. Optim. Sci. 39(1), 223–237 (2017). https://doi.org/10.1080/02522667.2017.1380420

Larman, C.: Agile and Iterative Development: A Manager’s Guide. Addison-Wesley Professional, Boston (2004)
Larman, C., Basili, V.: Iterative and incremental developments a brief history. Computer 36(6), 47–56 (2003). https://doi.org/10.1109/mc.2003.1204375

Littlewood, B.: Software reliability model for modular program structure. IEEE Trans. Reliab. 28(3), 241–246 (1979). https://doi.org/10.1109/tr.1979.5220576

Littlewood, B., Sofer, A.: A Bayesian modification to the Jelinski–Moranda software reliability growth model. Softw. Eng. J. 2(2), 30 (1987). https://doi.org/10.1049/sej.1987.0005

Luo, Z., Cao, P., Tang, G., Wu, L.: A modification to the Jelinski–Moranda software reliability growth model based on cloud model theory. In: 2011 Seventh International Conference on Computational Intelligence and Security. IEEE (2011). https://doi.org/10.1109/cis.2011.51

Mahapatra, G., Roy, P.: Modified Jelinski–Moranda software reliability model with imperfect debugging phenomenon. Int. J. Comput. Appl. 48(18), 38–46 (2012). https://doi.org/10.5120/ia190019

Mirjalili, S., Hashim, S.Z.M.: A new hybrid PSOGSA algorithm for function optimization. In: 2010 International Conference on Computer and Information Application. IEEE (2010). https://doi.org/10.1109/iccia.2010.6141614

Mirjalili, S., Wang, G.G., Coelho, L.D.S.: Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput. Appl. 25(6), 1423–1435 (2014). https://doi.org/10.1007/s00521-014-1629-6

Model, W.: Waterfall model. Luettavissa. 3 (2015). http://www.waterfall-model.com/Luettu

Moranda, P.B.: An error detection model for application during software development. IEEE Trans. Reliab. 30(4), 309–312 (1981). https://doi.org/10.1109/tr.1981.5221096

Musa, J.D., Okumoto, K.: A logarithmic poisson execution time model for software reliability measurement. In: Proceedings of the 7th International Conference on Software Engineering, pp. 230–238. Citeseer (1984)

Myung, I.J.: Tutorial on maximum likelihood estimation. J. Math. Psychol. 47(1), 90–100 (2003)

Ohba, M.: Inflection s-shaped software reliability growth model. In: Lecture Notes in Economics and Mathematical Systems, pp. 144–162. Springer, Berlin (1984). https://doi.org/10.1007/978-3-642-45587-2_10

Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Trans. Softw. Eng. 33(6), 402–419 (2007). https://doi.org/10.1109/tse.2007.1015

Ortiz, E.: Software-quality improvement using reliability-growth models. In: Annual Reliability and Maintainability Symposium 1995 Proceedings. IEEE (1995). https://doi.org/10.1109/rams.1995.513244

Pavlov, N., Iliev, A., Rahnev, A., Kyurkchiev, N.: Application of a new class cumulative lifetime distribution to software reliability analysis. Commun. Appl. Anal. 22(4), 555–565 (2018). https://doi.org/10.12732/caa.v22i4.4

Pham, H.: Software reliability modeling. In: Springer Series in Reliability Engineering, pp. 153–177. Springer, London (2006). https://doi.org/10.1007/1-84628-295-0_5

Rana, R., Staron, M., Berger, C., Hansson, J., Nilsson, M., Torner, F.: Comparing between maximum likelihood estimator and non-linear regression estimation procedures for NHPP software reliability growth modelling. In: 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement. IEEE (2013). https://doi.org/10.1109/iwsm-mensura.2013.37

Roy, P., Mahapatra, G.S., Dey, K.N.: AN NHPP software reliability growth model with imperfect debugging and error generation. Int. J. Reliab. Qual. Saf. Eng. 21(02), 1450008 (2014). https://doi.org/10.1142/s0218393314500089

Sangeeta, Sitender: Comprehensive analysis of hybrid nature-inspired algorithms for software reliability analysis. J. Stat. Manag. Syst. 23(6), 1037–1048 (2020)

Sangeeta, Sharma, K., et al.: A quantitative testing effort estimate for reliability assessment of multi release open source software systems. J. Comput. Theor. Nanosci. 16, 5089–5098 (2019)

Sangeeta, Sharma, K., Bala, M.: An ecological space based hybrid swarm-evolutionary algorithm for software reliability model parameter estimation. Int. J. Syst. Assur. Eng. Manag. 11(1), 77–92 (2020a)

Sangeeta, Sharma, K., Bala, M., et al.: Reliability analysis and modeling of green computing based software systems. Recent Adv. Comput. Sci. Commun. https://www.researchgate.net/publication/
Schick, G., Wolverton, R.: An analysis of competing software reliability models. IEEE Trans. Softw. Eng. SE-4(2), 104–120 (1978). https://doi.org/10.1109/tse.1978.231481

Sharma, K., Garg, R., Nagpal, C.K., Garg, R.K.: Selection of optimal software reliability growth models using a distance based approach. IEEE Trans. Reliab. 59(2), 266–276 (2010). https://doi.org/10.1109/tr.2010.2048657

Sharma, K., Bala, M., et al.: Magnetic navigation based optimizer: a new optimization algorithm for software reliability model parameter estimation. J. Adv. Res. Dyn. Control Syst. (14), 1957–1968 (2018)

Sheta, A.: Reliability growth modeling for software fault detection using particle swarm optimization. In: 2006 IEEE International Conference on Evolutionary Computation IEEE (2006). https://doi.org/10.1109/cec.2006.1688697

Singh, V., Sharma, M., Pham, H.: Entropy based software reliability analysis of multi-version open source software. IEEE Trans. Softw. Eng. 44(12), 1207–1223 (2018). https://doi.org/10.1109/tse.2017.2766070

Smidts, C., Stutzke, M., Stoddard, R.: Software reliability modeling: an approach to early reliability prediction. IEEE Trans. Reliab. 47(3), 268–278 (1998)

Tripathi, A.K., Sharma, K., Bala, M.: Military dog based optimizer and its application to fake review. arXiv preprint arXiv:190911890 (2019)

Turk, L.I.A., Alsolami, E.G.: Jelinski–Moranda software reliability growth model?: A brief literature and modification. Int. J. Softw. Eng. Appl. 7(2), 33–44 (2016). https://doi.org/10.5121/ijsea.2016.7204

Wahab, M.N.A., Netfi-Meziani, S., Ayabi, A.: A comprehensive review of swarm optimization algorithms. PLOS ONE 10(5), e0122827 (2015). https://doi.org/10.1371/journal.pone.0122827

Wang, W.L., Wu, Y., Chen, M.H.: An architecture-based software reliability model. In: Proceedings: Pacific Rim International Symposium on Dependable Computing. IEEE Computer Society (1999). https://doi.org/10.1109/prdc.1999.816223

Wei, X., Dong, Y., Li, X., Wong, W.E.: Architecture-level hazard analysis using AADL. J. Syst. Softw. 137, 580–604 (2018). https://doi.org/10.1016/j.jss.2017.06.018

Wei, S., Weyuker, E.: An extended domain-based model of software reliability. IEEE Trans. Softw. Eng. 14(10), 1512–1524 (1988). https://doi.org/10.1109/32.6196

Wong, W.E., Gao, R., Li, Y., Abreu, R., Wotawa, F.: A survey on software fault localization. IEEE Trans. Softw. Eng. 42(8), 707–740 (2016). https://doi.org/10.1109/tse.2016.2521368

Xavier, J., Macêdo, A., Matias, R., Borges, L.: A survey on research in software reliability engineering in the last decade. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM Press (2014). https://doi.org/10.1145/2554850.2555161

Xiang, J., Machida, F., Tedano, K., Maeno, Y.: An imperfect fault coverage model with coverage of irrelevant components. IEEE Trans. Reliab. 64(1), 320–332 (2015). https://doi.org/10.1109/tr.2014.2363155

Xie, M.: Software Reliability Modelling, vol. 1. World Scientific, Singapore (1991)

Yamada, S., Ohba, M., Osaki, S.: S-shaped reliability growth modeling for software error detection. IEEE Trans. Reliab. R-32(5), 475–484 (1983). https://doi.org/10.1109/tr.1983.5221735

Yamada, S., Ohtera, H., Narihisa, H.: Software reliability growth models with testing-effort. IEEE Trans. Reliab. 35(1), 19–23 (1986). https://doi.org/10.1109/tr.1986.4335332

Yamada, S., Ohtera, H., Ohba, M.: Testing-domain dependent software reliability models. Comput. Math. Appl. 24(1–2), 79–86 (1992)

Yang, X.S.: Nature-inspired matheuristic algorithms: success and new challenges. J. Comput. Eng. Inf. Technol. 01(01), 1–3 (2012). https://doi.org/10.4172/2324-9307.1000e101

Yang, X.S.: Bat algorithms. In: Nature-Inspired Optimization Algorithms, pp. 141–154. Elsevier, Amsterdam (2014). https://doi.org/10.1016/b978-0-12-416743-8.00010-5

Zhang, X., Teng, X., Pham, H.: Considering fault removal efficiency in software reliability assessment. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. 33(1), 114–120 (2003). https://doi.org/10.1109/tsmca.2003.812597

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
Authors and Affiliations

Sangeeta¹,² · Sitender³ · Kapil Sharma⁴ · Manju Bala⁵

Sangeeta
sangeeta.phogat@gmail.com; sangeeta@msit.in

Kapil Sharma
kapil@ieee.org

Manju Bala
manjugpm@gmail.com

¹ Department of Computer Science and Engineering, Delhi Technological University, Rohini, Delhi, India
² Present Address: Department of Computer Science and Engineering, Maharaja Surajmal Institute of Technology, Janakpuri, New Delhi 110058, India
³ Department of Information Technology, Maharaja Surajmal Institute of Technology, New Delhi 110058, India
⁴ Department of Information Technology, DTU, Rohini, Delhi, India
⁵ Department of Computer Science, Inderprastha College for Women, DU, Civil Lines, Delhi, India